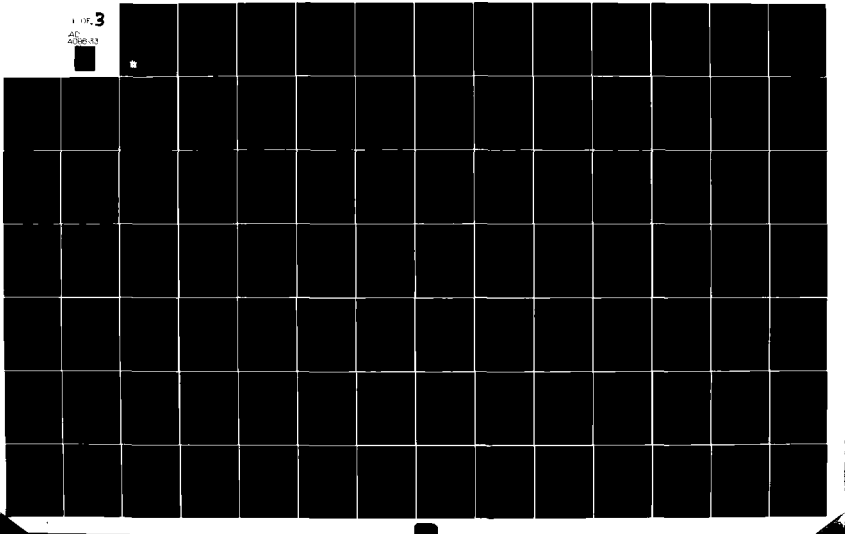
AD-A086 133 NOTRE DAME UNIV IN DEPT OF ELECTRICAL ENGINEERING F/G 17/2
DESIGN AND IMPLEMENTATION OF A SPEECH CODING ALGORITHM AT 9600 --ETC(U)
APR 80 J L MELSA, D L COHN, A ARORA DCA100-79-C-0005

UNCLASSIFIED NL

1 OF 3
AD A086133



LEVEL I

12

FINAL REPORT
VOLUME 1

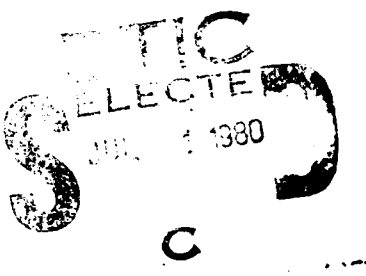
DESIGN AND IMPLEMENTATION OF A SPEECH CODING ALGORITHM AT 9600 B/S

Department of
ELECTRICAL ENGINEERING
UNIVERSITY OF NOTRE DAME, NOTRE DAME, INDIANA

12

FINAL REPORT
VOLUME 1

DESIGN AND IMPLEMENTATION OF A SPEECH CODING ALGORITHM AT 9600 B/S



Prepared for

Defense Communications Agency
Defense Communications Engineering Center
1860 Wiehle Avenue
Reston, Virginia 22090

Contract No. DCA 100-79-C-0005

30 April 1980

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO. AD-A086 133	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Design and Implementation of a Speech Coding Algorithm at 9600 B/S	5. TYPE OF REPORT & PERIOD COVERED Final Report Nov. 1978 - April 1980
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) James L. Melsa, et al.	8. CONTRACT OR GRANT NUMBER(s) DCA 100-79-C-0005
9. PERFORMING ORGANIZATION NAME AND ADDRESS Department of Electrical Engineering University of Notre Dame Notre Dame, IN 46556	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Communications Agency Contract Management Division, Code 260 Washington, D.C. 20305	12. REPORT DATE April 1980
	13. NUMBER OF PAGES 519
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)	15. SECURITY CLASS. (of this report) Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution of this document is unlimited. It may be released to the Clearinghouse, Department of Commerce, for sale to the general public.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Speech coding, 9600 bps speech transmission, pitch extraction, adaptive residual coder, waveform reconstruction.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report describes a speech coding algorithm for digital transmission of speech at a rate of 9600 bits per second and the implementation of this algorithm on a speech processing system. The algorithm combines:

Pitch extraction loop,
Pitch compensating adaptive quantizer,
Sequentially adaptive linear predictor,
Adaptive source coding,

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

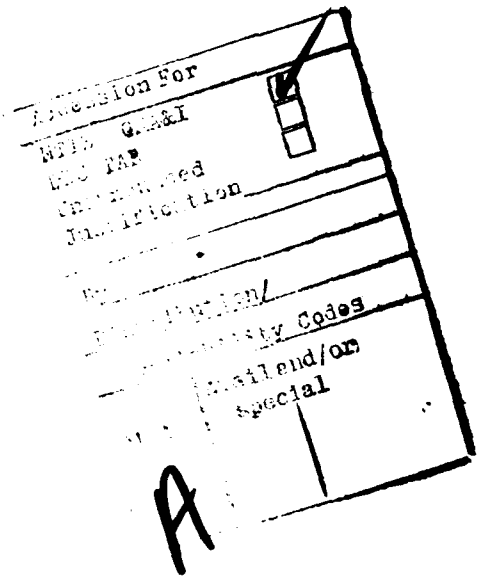
S/N 0102-014-6601

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Ent

to generate very high quality speech output. Although each of these elements has been previously applied to speech coding, the combination of all four of these elements has not been studied before. The speech coding algorithm has been implemented on a pair of CSPI MAP 300 Array Processors in real-time in the full-duplex mode.

This report has been bound in two volumes. The first volume contains the narrative description of the algorithm and its development and includes Chapters 1 through 11 and Appendices A through D of the report. The second volume describes the real-time MAP implementation and includes Chapters 12 and Appendices E through G.



ABSTRACT

This report describes a speech coding algorithm for digital transmission of speech at a rate of 9600 bits per second and the implementation of this algorithm on a speech processing system. The algorithm combines

- Pitch extraction loop
- Pitch compensating adaptive quantizer
- Sequentially adaptive linear predictor
- Adaptive source coding

to generate very high quality speech output. Although each of these elements has been previously applied to speech coding, the combination of all four of these elements has not been studied before. The speech coding algorithm has been implemented on a pair of CSPI MAP 300 Array Processors in real-time in the full-duplex mode.

This report has been bound in two volumes. The first volume contains the narrative description of the algorithm and its development and includes Chapters 1 through 11 and Appendices A through D of the report. The second volume describes the real-time MAP implementation and includes Chapters 12 and Appendices E through G.

PROJECT PERSONNEL

Arvind Arora, research assistant

David L. Cohn, co-principal investigator

James M. Kresse, research assistant

James L. Melsa, principal investigator

Arun K. Pande, research assistant

Maw-lin Yeh, research assistant

FINAL REPORT

DCA CONTRACT 100-79-C-0005

	Page
Abstract	i
Project Personnel	ii
1. Introduction and Outline of Report	1
1.1 Introduction	1
1.2 Summary of Algorithm Requirements	3
1.3 Outline of Report	4
2. Algorithm Description	5
2.1 Introduction	5
2.2 Transmitter Buffer System	11
2.3 Adaptive Low-Pass Filter	16
2.4 Pitch Extraction	18
2.5 Adaptive Residual Coder	20
2.5.1 Adaptive Predictor	20
2.5.2 Adaptive Quantizer	22
2.6 Pitched Repetition	26
2.7 Noiseless Source Coder	28
2.8 Receiver	34
3. Synchronization	38
3.1 Sync. Algorithm Description	40
3.1.1 Synchronization Acquisitions	40
3.1.2 Synchronization Monitor	43
4. Pitch Extraction Studies	46
4.1 Introduction	46
4.2 Pitch Extraction Algorithms	47
4.3 Redundancy Removal	54
4.4 References	61
5. Tree Coding	62
5.1 Introduction	62
5.2 The (M,L) Algorithm	63
5.2.1 Description	63
5.2.2 Results	65
5.3 Adaptive Tree Coding	69
5.3.1 Description	69
5.3.2 Results	71
5.4 Conclusions and Suggestions for Further Research	75
5.5 References	77

6. Backward PARC 78

6.1 Introduction 78
6.2 System Structure 80
6.3 Performance Evaluation and Parametric Studies 85
6.4 Transmission Error Studies 90
6.5 Conclusions 95

7. Tandem Operation 96

7.1 Introduction 96
7.2 PARC in Tandem with CVSD 97
7.3 Effect of Background Noise on PARC Performance 100
7.4 References 103

8. Transmission Errors 104

8.1 Introduction 104
8.2 Simulation of Transmission Error 105
8.3 Minimizing Transmission Error Effects 108
8.4 Conclusion 112
8.5 Reference 113

9. Filtering 114

9.1 Introduction 114
9.2 Evaluation of Pre-emphasis 115
9.3 Filter Selection 118
9.4 Results 121
9.5 Low-Pass Filtering vs. Entropy 123
9.6 Results and Conclusions 125
9.7 References 128
9.A Derivation of Filter 129

10. Buffer Control 136

10.1 Introduction 136
10.2 Overflow control 137
10.3 Underflow control 138
10.4 Special considerations at the receiver 139
10.5 Conclusions and suggestions for further research 140

11. Source and Error Control Coding 141

11.1 Introduction 141
11.2 Source Coding 142
11.2.1 Quantizer levels 142
11.2.2 Side information 144
11.3 Error Control Coding 145
11.4 Conclusion and suggestions for further research 146

A. FORTRAN Simulation of Algorithm 147

B. Segmented SNR Plots 198
B.1 Introduction 198
B.2 Listings 199

C. Plotting Program 242

D. PDP-11 D/A Programs 247

12. MAP Implementation 263

12.1 Introduction 263
12.2 PARC - Transmitter 278
12.2.1 The Pitch Computation Program 281
12.2.2 The Speech Digitization Program 288
12.3 PARC - Receiver 295
12.4 Noiseless Source Coder 302
12.5 Synchronizer, Decoder 311
12.5.1 Synchronization Acquisition Module 313
12.5.2 Decoder Module 315
12.5.3 Initialization Module 318
12.5.4 Decoder Constraints 319
12.6 Program Timing and Speed 320

E. Resampling Program 322
E.1 Operating Details 325

F. CVSD Algorithm 344
F.1 The CVSD System 344
F.2 The Real-Time CVSD System 346
F.2.1 Design Overview 346
F.2.2 The Initialization Step 348
F.2.3 The Processing Step 351
F.3 Conclusion 355
F.4 Reference 355

G. Program Listings for Real Time Implementation of PARC on MAP-300 370

CHAPTER 1
INTRODUCTION AND OUTLINE OF REPORT

1.1 Introduction

This report describes the results of a sixteen month effort under DCA Contract 100-79-C-0005 to develop and implement a speech coding algorithm designed to produce very good quality speech at 9600 bits per second. The technique is based on the latest developments in speech digitization and is formulated to comply with the requirements described in the statement of work.

The method is based on techniques which have been reported in the literature but which are brought together here for the first time. The system combines elements of adaptive predictive coding and ADPCM systems and is known as PARC, Pitch extraction Adaptive Residual Coder. The pitch extraction loop used in adaptive predictive coding provided the input to a sequentially adaptive predictor, using backward coefficient adaptation which forms an estimate of the pitch-reduced signal. The error in this estimate is quantized by a pitch compensating adaptive quantizer. The resulting quantizer output is coded using an adaptive source coding procedure. The source code also permits transmission of pitch information and synchronization signals.

This algorithm was implemented on a pair of CSPI MAP 300 signal processors to generate a real-time, full-duplex speech encoding system.

This algorithm has several features which are significant in a full system application. The waveform reconstruction nature of the algorithm provides excellent performance in tandem with CVSD and in the presence of background noise. If bit rates higher than 9600 b/s are permitted,

the algorithm is easily adaptive to them. For example, a 16 kb/s version of the same algorithm will differ only in the number of quantization levels and the source code algorithm. It could easily be implemented using the same software.

1.2 Summary of Algorithm Requirements

The following requirements for the speech coding algorithm have been determined from the Statement of Work.

1. The speech processing system shall operate a transmission data rate of 9600 b/s.
2. The speech processing system shall produce very high quality speech reproduction. This requirement is interpreted to mean a signal-to-noise ratio of approximately 20 db.
3. The audio bandwidth of the speech coder shall be greater than or equal to 3200 Hz.
4. The speech coder shall produce good quality speech under conditions of a random transmission bit error rate of 1 percent.
5. The speech coder shall produce intelligible speech under conditions of acoustic background noise (60 db referenced to 20 μ Newton/meter^2) such as office noise.
6. The speech coder shall perform satisfactorily in tandem with a CVSD speech coder operating at a data rate of 16 kb/s. This tandem configuration shall provide speech intelligibility with minimal degradation compared with a single link of CVSD operating at 16 kb/s.

The algorithm shall be implemented on a pair of CSPI MAP 300 signal processor in real-time, full duplex mode with appropriate synchronization.

1.3 Outline of Report

Following the introductory material in this chapter, the next two chapters of the report describe the general details of the PARC algorithm. Chapter 2 describes the final form of the PARC speech digitization algorithm developed in this study.

The following nine chapters of the report describe in more depth the details of the algorithm and various studies that were made during this contract period but which do not necessarily appear in the final algorithm. Chapter 3 delineates the details of synchronization for full duplex operation. Chapter 4 describes various pitch extraction studies, while Chapter 5 describes details of the tree coding studies that were conducted. A special form of the PARC algorithm in which backward adaptation is used for the pitch extraction operation is described in Chapter 6. The operation of the algorithm in tandem with CVSD is discussed in Chapter 7, while Chapter 8 is concerned with transmission studies. The final algorithm uses an adaptive filter on the input speech to improve subjective performance. This algorithm is described in Chapter 9. Chapter 10 is concerned with the buffer control algorithms used in the PARC system, while Chapter 11 is concerned with source and error control coding for full-duplex operation.

Chapter 12 describes the real-time implementation of the algorithm on the MAP processor. The remainder of the report is a series of appendices which describe various details of the programming of the algorithms in both Fortran and on the MAP, as well as various support packages.

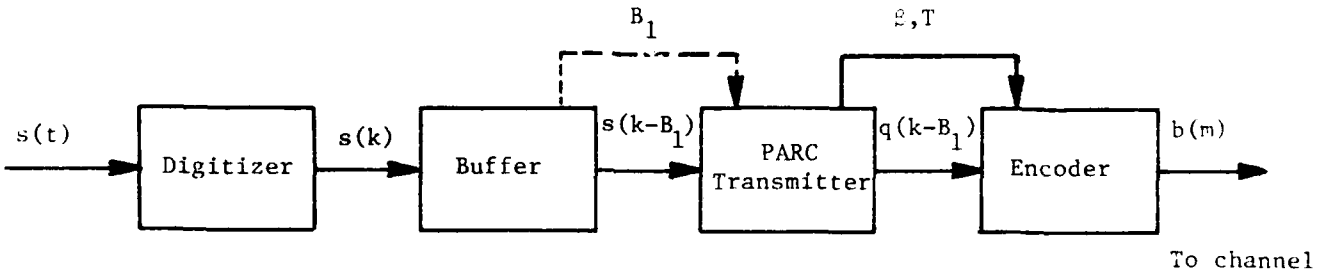
CHAPTER 2
ALGORITHM DESCRIPTION

2.1 Introduction

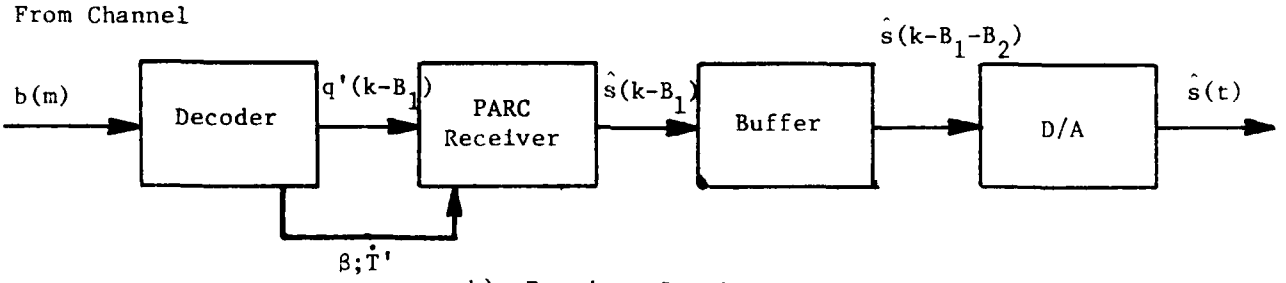
This chapter will describe the final form of the PARC speech digitization algorithm developed during this study. The algorithm will be decomposed into its constituent elements and each element will be described in turn. The underlying theory will be discussed and the details of the recommended implementation will be presented. Later chapters will describe the real-time implementation and the various studies which led to the recommended form of the algorithm.

The PARC digital speech communication system can be represented as shown in Fig. 2.1. The analog speech signal s(t) is converted to a sequence of finely quantized samples s(k). These quantized samples are stored in a buffer whose delay B_1 can vary over time. The PARC transmitter section does the actual data reduction to produce the quantizer level sequence. This is represented by the $q(k-B_1)$. This sequence, along with the side information quantized β and T, is noiselessly encoded into the bit stream b(m) for transmission on the channel.

At the receiver end, the process is essentially reversed. The bit stream b'(m) is decoded into the sequence $q'(k-B_1)$ and the side information β' and T'. The primes are used to allow for channel errors. The PARC receiver device converts this information into reconstructed speech $\hat{s}(k-B_1)$. This is buffered with a variable delay B_2. A D/A output unit presents a filtered version of the delayed speech to the user. The overall system delay B_1+B_2 is a constant.



a) Transmitter Section



b) Receiver Section

Fig. 2.1 Speech Digitization System

The analog filters were designed by GTE under a separate contract. They are described elsewhere. Therefore, the discussion here will begin after the continuous signal has been converted to s(k).

Figure 2.2 shows a more detailed block diagram of the transmitter system. The system consists of the following major components:

- SAMPLE buffer
- Adaptive Low-pass filter
- Pitch extraction loop
- Adaptive Residual Coder
- Noiseless source coder.

As noted, the SAMPLE buffer receives the incoming speech samples and holds them for further processing. For notational simplicity, the samples at the output of this buffer will be referred to as s(k). The adaptive low-pass filter is used as needed to help prevent an overflow of the SAMPLE buffer. The output of the adaptive low-pass filter $s_f(k)$ forms the input to the pitch extraction loop. The pitch extraction loop uses a block of these filtered samples $s_f(k)$ to estimate the pitch period T and the correlation coefficient β. Using this information, the pitch-reduced speech samples v(k) are calculated by

$$v(k) = s_f(k) - \beta\hat{s}(k-T) \qquad (2.1)$$

The pitch-reduced speech samples v(k) are then processed by the Adaptive Residual Coder. An estimate p(k) of v(k), produced by the adaptive predictor is subtracted from v(k) to form the prediction error e(k). The prediction error e(k) is then passed through an adaptive quantizer to yield the quantizer level q(k). This quantizer level q(k) is the

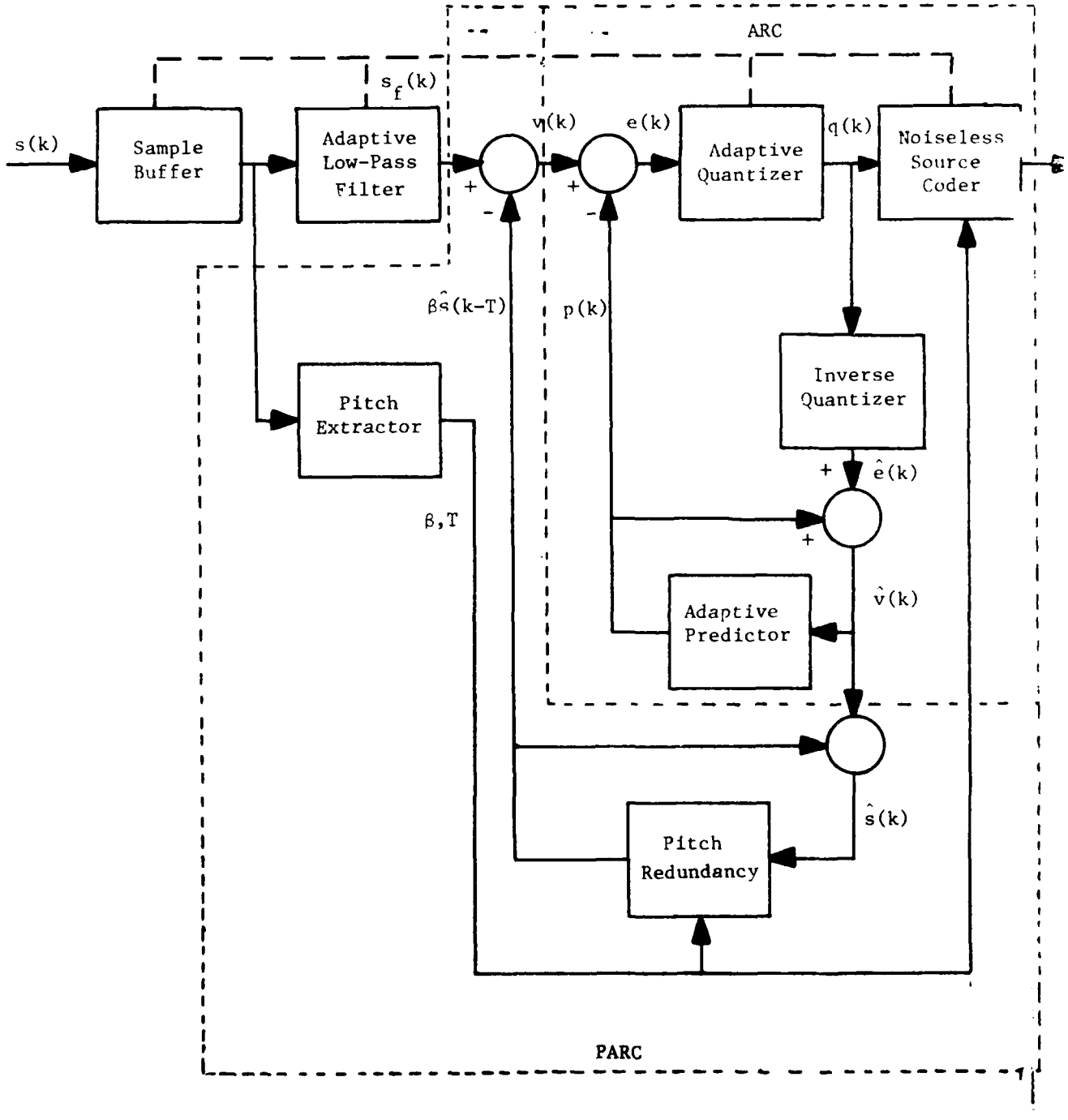


Fig. 2.2 PARC Transmitter

input to both the inverse quantizer (to update the rest of the system) and the noiseless source coder (to be transmitted down the channel). The noiseless source coder combines the quantizer level information q(k), the pitch period T, and the correlation coefficient β, and generates the binary bit stream b(m) to convey this information to the receiver.

The underlying design principle of the adaptation procedure is that all information used in updating the inverse quantizer and the predictor be available both at the transmitter and the receiver. This will allow the receiver to replicate these devices. Since the only information sent from the transmitter to the receiver is the quantizer output and pitching information, the adaptation procedures for the inverse quantizer and the predictor must use quantities derivable from them and a from pre-arranged initial state.

Although the PARC is basically a sequential system, the use of pitch redundancy reduction forces a block structure. For each block, new values of β and T are computed. The resulting block structure appears throughout the real-time implementation of the system.

Subsequent sections will describe each of the subsystems in the transmitter. The next section explains the operation of the transmitter buffer system. Since the noiseless source coder produces a variable number of bits for each sample, the rate at which samples are processed varies with time. Thus, the buffering operation is quite complex and important.

Section 2.3 describes the implementation of the adaptive low-pass filter. The filter is only activated when the SAMPLE buffer is almost full and it is only used to reduce the rate at which bits are generated. Thus, its operation is closely related to that of the SAMPLE buffer.

full and it is only used to reduce the rate at which bits are generated. Thus, its operation is closely related to that of the SAMPLE buffer.

The pitch extraction algorithm is a fairly standard AMDF system and is explained in Section 2.4. Although β and T are computed for a fixed number of speech samples, the number of samples they are actually used on will vary.

The adaptive residual coder is a sophisticated ADPCM system. Section 2.5 describes parametric modifications used to optimize the structure for pitch-reduced speech and to combat channel errors. A new feature, known as pitched repetition, has been added to the ARC to reduce the bit rate during voiced speech. It is described in Section 2.6.

A central feature of the PARC system is the noiseless source coding structure. It allows the most efficient use of all channel bits to yield the highest fidelity speech. In the final implementation, it includes the binary representation of q(k), β and T as well as synchronization and error control. Section 2.7 gives the details of this component.

The next-to-last section of this chapter describes the structure of the receiver. It too has a complex buffer structure which must be considered during the system design. The final section is a list of references to the various techniques employed by PARC.

2.2 Transmitter Buffer System

The buffering of data flows in the transmitter is a fairly complex procedure. Due to the variable rate coding procedure, the overall delay experienced by a speech sample traveling through the transmitter varies with time. During unvoiced speech or silence, the delay will be short but during voiced speech, it can be over a thousand sample times. However, the receiver must produce one reconstructed speech sample for each sample that enters the transmitter. Thus, if there are no channel errors, the overall system delay is fixed.

There are actually four separate buffers used in the transmitter buffer system. Three are used to facilitate data transfer and to allow parallel processing; one accommodates the variable delay. The buffers are the ADAM buffer, the SAMPLE buffer, the LEVEL buffer and the BIT buffer. They are shown schematically in Fig. 2.3. The ADAM buffer, the LEVEL buffer, and the BIT buffer are each double buffers which allow parallel data processing and have essentially a fixed delay. The SAMPLE buffer, though, is a large circular buffer which accommodates the system's variable-rate encoding.

The operation of the four buffers is best described by first explaining the input and output for each of them. Continuous speech signal enters the PARC system through an analog speech interface. This interface conditions the signal for the Analog Data Acquisition Module (ADAM) by low-pass filtering it and by adjusting the signal level to the range of the analog-to-digital converter in the ADAM. The ADAM samples this signal at a rate of 6.4 Kss and places the samples into the ADAM buffer. Thus, the input to the buffer is a sequence of single speech samples at the rate of one each 156.25 μsec.

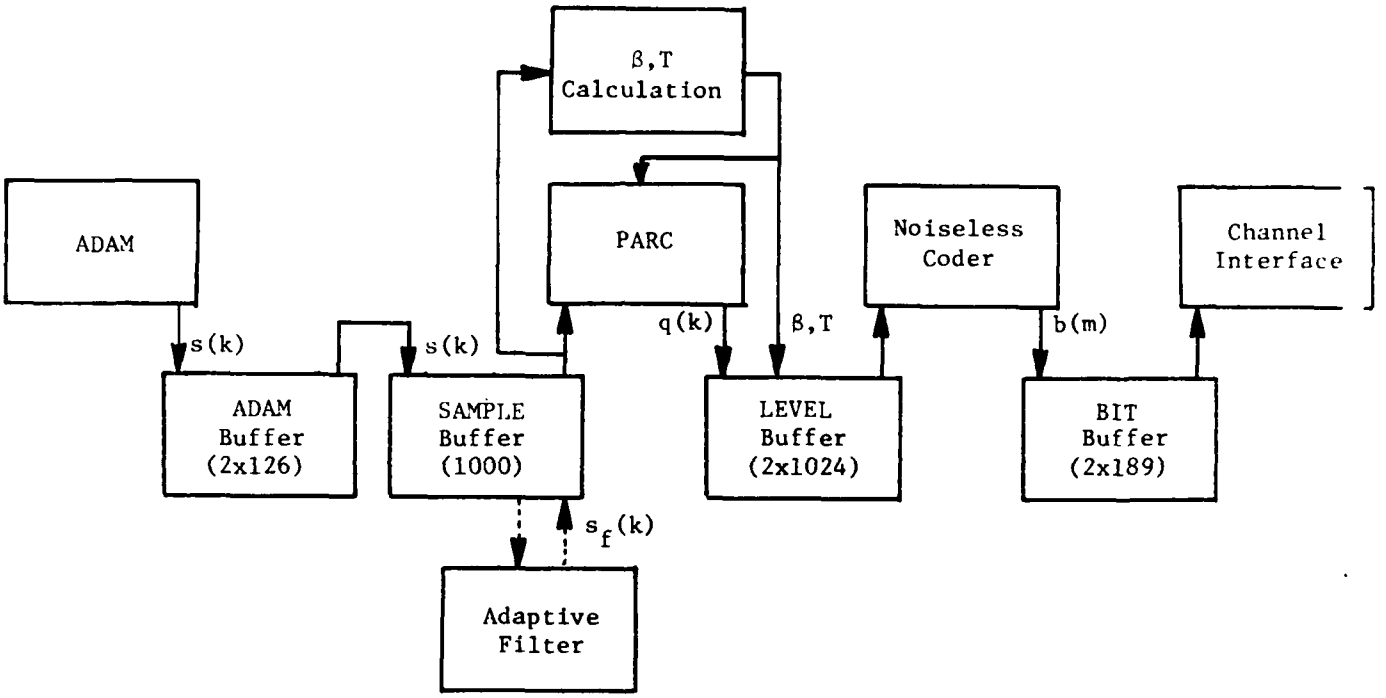


Fig. 2.3 Transmitter Buffer System

Data is removed from the ADAM buffer in blocks of 126 samples each 19 687.5 μsec. Thus, data enters and leaves the buffer at an average rate of 6400 samples per second. The blocks of 126 samples are transferred to the SAMPLE buffer. The rate at which these samples are processed by the noise reducer and then are removed from the SAMPLE buffer depends on the number of bits they generate. The block-time of 19 687.5 μsec. was selected to correspond to both 126 input sampling intervals and to the time allowed to transmit 189 bits on the channel. Thus, the number of samples N_B removed from the SAMPLE buffer in one block-time will be the number which generates at least 189 channel bits. This can range from as many as 500 or more during silence to fewer than 80 during voiced speech. The SAMPLE buffer must be large enough to accommodate this kind of variation without introducing undue delay.

The samples removed from the SAMPLE buffer in a given block time are processed by the PARC algorithm and the corresponding quantizer levels q(k) are generated. The N_B samples will generate N_B quantizer levels. These are placed in the LEVEL buffer. Since PARC operates on a sample-by-sample basis, the q(k) are loaded into their buffer one at a time.

The source coder generates full blocks of 189 bits. Therefore, it removes N_B quantizer levels from the LEVEL buffer at one time. The resulting block of bits are stored in the bit buffer. They are clocked out of this buffer at the rate of 9600 bits per second.

The ADAM buffer, as all double buffers, actually has two half-buffers, each of which holds 126 speech samples. While the ADAM is filling one half-buffer with the incoming samples, the other half-buffer can be emptied into the SAMPLE buffer. The SAMPLE buffer is a circular buffer which holds up to 1024 speech samples. There are two pointers associated with the buffer which keep track of where samples are to enter the buffer

and from where they are to leave the buffer. The distance between the pointers indicates how many samples are currently in the sample buffer.

Before the samples are removed from the SAMPLE buffer, three operations are performed. If the SAMPLE buffer is over a pitched repetition threshold, it will signal the transmitter to do the pitched repetition as described in Section 2.6. If the adaptive low-pass filter is called for, the oldest 80 samples in the buffer are filtered. The original samples are replaced with the filtered samples. The β and T values for de-pitching are then calculated based on the 80 oldest samples in the buffer. Samples are then passed to the PARC for processing and the corresponding quantizer levels are put into the LEVEL buffer. For each sample, the number of information bits required to represent the quantizer level is computed. The process will stop at either of the following three conditions:

1. The transmitter has generated the required 157 information bits.
2. The transmitter has no more input samples to process, i.e., the SAMPLE buffer is in an underflow condition. In this case, the transmitter will output a NULL code to the source encoder.
3. The transmitter has processed the maximum number of samples allowed to be in real-time.

The LEVEL buffer is a large double buffer capable of holding up to 1200 quantizer levels. Even though the buffer can hold 1200 levels, it will never have more than two blocks each of which generates 157 information bits. One half-buffer is used for incoming quantizer levels, while the other half-buffer is available to the noiseless source coder. The noiseless source coder takes these quantizer levels, and, with the associated quantized β and T, generates a 189-bit block which is placed in the BIT buffer.

The BIT buffer is a double buffer which holds 378 bits. One half-buffer receives bits from the channel coder, while the other half-buffer is available for output to the channel. Output takes place through the Input/Output Scroll (IOS-2). The IOS-2 provides the bits to the modem interface contained in the speech interface unit.

2.3 Adaptive Low-Pass Filter

Adaptive low-pass filtering is used in the system to provide a soft-failure capability under certain circumstances. When the SAMPLE buffer fills more rapidly than it is being emptied, for example during voiced speech, it is possible that it could overflow. It was found that low-pass filtering the speech mitigated this problem. When the SAMPLE buffer is nearly full, therefore, the speech is low-pass filtered to help guard against buffer overflow.

The recommended low-pass filter has been derived from a first order Butterworth filter, using the bilinear transformation to obtain a digital filter. The general form of the transfer function of this type of filter is

$$H(z) = \frac{\omega_{CA} + \omega_{CA} z^{-1}}{(\omega_{CA}+1) + (\omega_{CA}-1)z^{-1}} \qquad (2.2)$$

where

$$\omega_{CA} = \tan \frac{\pi f_c}{f_s}$$

f_C = cutoff frequency in Hz.

f_S = sampling frequency in Hz.

The selected parameters are f_C = 1800Hz and f_S = 6400 Hz. Transforming back into the sample domain, the filter equation can be written as

$$s_f(k) = As(k) + As(k-1) + Bs_f(k-1) \qquad (2.3)$$

where

$$A = \frac{-\sqrt{2}C_A}{C_A + 1}$$

$$B = \frac{C_A - 1}{C_A + 1}$$

The adaptive low-pass filtering operates by checking the number of samples in the SAMPLE buffer just prior to the pitch extraction calculations. If there are fewer than 501 samples in the SAMPLE buffer at that time, the low-pass filtering is skipped. If there are 501 or more samples in the same buffer, the filtering is implemented. Thus, filtering is only used when buffer overflow is threatened.

When filtering is called for, a block of 80 samples are low-pass filtered and the original samples are replaced by the filtered samples. A block of N_B samples are then processed by PARC. Note that N_B may be less than 80 so that some filtered samples may remain in the buffer and may be filtered again during the next block time.

The filtering is performed in blocks of 80 for two reasons. First, filtering in large blocks reduces the number of transitions between filtered and unfiltered speech. Second, the larger block causes some samples to be multiple-filtered if the sample buffer continues to fill. The design of the filter causes the effect of this multiple-filtering to be similar to filtering with a lower cutoff frequency. For the recommended parameters, using the 1800 Hz cutoff frequency filter twice results in an overall filter with a cutoff frequency of about 1350 Hz. In this way, the low-pass filtering is automatically increased when needed.

2.4 Pitch Extraction

It is well known that voiced speech is highly correlated from pitch period to pitch period. A long-term prediction of $s_f(k)$ given by

$$s(k|k-T) = \beta s_f(k-T) \tag{2.4}$$

can be a good approximation to $s_f(k)$ for proper choice of β and T. The optimum scale factor β depends on the correlation between $s_f(k)$ and $s_f(k-T)$, and the best T is an estimate of the pitch period measured in samples. The goal in selecting β and T is to minimize the time average prediction error over a block of K samples

$$E = \frac{1}{K} \sum_{j=1}^{K} [s_f(j) - \beta s_f(j-T)]^2 \tag{2.5}$$

In most applications, β and T are computed and used over a given block of K samples. This procedure is modified in PARC. New values of β and T are computed each block-time. Therefore, they are held constant for N_B samples and, it will be recalled, N_B varies from block to block. However, it is not possible to determine N_B until after β and T have been chosen. Therefore, β and T are always calculated for a fixed block size and used for a variable number of samples. In the recommended implementation, the fixed computation block is K = 80.

The pitch period estimate T is computed by forming the Average Magnitude Difference Function (AMDF) and picking the value of T for which this function is minimized. The AMDF function is

$$A(T) = \sum_{j=1}^{K} |s_f(j) - s_f(j-T)| \tag{2.6}$$

In this way, the value of T selected usually matches the value obtained by minimizing the error E in Eq. (2.5) but with far fewer computations. Once T has been found, the error E is minimized by selecting β according to

$$\beta = \frac{\sum_{j=1}^{K} s_f(j)s_f(j-T)}{\sum_{j=1}^{K} s_f^2(j-T)} \qquad (2.7)$$

For some blocks of speech, such as those including transition regions from silence or unvoiced speech to voiced speech, the value of β given by Eq. (2.7) can be large. The use of such large values, however, can actually decrease the overall performance. Therefore, β was limited to the range of [-2,2]. The β's were uniformly quantized over this range. It was found that system performance was relatively insensitive to quantization noise on β. Therefore, β is quantized to 97 levels. This is represented in the transmitted block with seven bits. The 31 patterns of these seven bits which do not represent valid β values are used for another purpose.

By extracting pitch from different sentences and different speakers, it was found that the pitch period T varied between 24 and 70 samples. Hence, the searching range was chosen to be between 20 and 83 yielding 64 possible values of T, which requires an 8-bit codeword for T.

2.5 Adaptive Residual Coder

The heart of the PARC system is an Adaptive Residual Coder (ARC). The version recommended here has been modified to optimize its operation as a part of PARC. The input to the ARC is the pitch-reduced speech.

$$v(k) = s_f(k) - \beta\hat{s}(k-T) \tag{2.10}$$

where $\hat{s}(k-T)$ is the reconstructed version of $s_f(k-T)$ available at both the transmitter and receiver. The ARC consists of two principal subsystems: an adaptive predictor and an adaptive quantizer. These will be described in separate subsections.

2.5.1 Adaptive Predictor

The adaptive predictor produces a linear prediction p(k) given by

$$p(k) = \sum_{i=1}^{N} a_i(k)\ \hat{v}(k-i) \tag{2.11}$$

which is to be an estimate of v(k). The $\hat{v}(k-i)$ are the receiver's estimate of v(k-i). It can be argued that the predictor order N should match the order of the system which generates the v(k). However, predictors of order larger than 4 yield unsatisfactory performance in the presence of channel errors. Therefore, N=4 is used.

If the $a_i(k)$ accurately model the v(k), and if the $\hat{v}(k-i)$ are close to the v(k-i), then p(k) will be a good approximation to v(k). The $a_i(k)$ are adaptive, and after p(k) is formed, they are updated. They are adapted according to steepest descent of $e^2(k)$. This is approximated in the system by the following updating algorithm:

$$a_i(k+1) = \delta b_i + (1-\delta)\ \left[a_i(k) + \frac{g\ \hat{v}(k-i)\ \hat{e}(k)}{<|\hat{v}(k)|>^2}\right] \tag{2.12}$$

where $<|\hat{v}(k)|>$ is a biased exponential time average of $|\hat{v}(k)|$

$$<|\hat{v}(k)|> = (1-\alpha) \sum_{j=0}^{\infty} \alpha^j |\hat{v}(k-j)| + \text{RMSMIN}$$

Thus, the $a_i(k)$ updating algorithm has eight parameters: δ, g, α, RMSMIN and b_i for i=1,2,3 and 4. Three of them, δ, g and α, essentially determine how much memory there is in the updating process. In order to minimize the effect of channel errors, the memory time was reduced from what would be optimal in the error-free case. This did not significantly degrade performance. The recommended values of these parameters are

$$\delta = 0.01$$

$$g = 0.02$$

$$\alpha = 0.90$$

The parameters b_i represent the quiescent values of the coefficients $a_i(k)$. The values used in the original ARC are also recommended here.

$$b_i = \begin{cases} 0.7 & i=1 \\ 0 & i=2,\ 3 \text{ or } 4 \end{cases}$$

The quantity RMSMIN is perhaps the most sensitive parameter in the algorithm. It determines the minimum value of $<|\hat{v}(k)|>$ which affects both the adaptive predictor and the adaptive quantizer. The lower RMSMIN, the more the system responds during low level signals. This reduces granular noise and increases the data rate. The higher data rate means that the sample buffer fills faster leading to more low-pass filtering and increased use of pitched repetition. The value selected for RMSMIN must be matched to the dynamic range of s(k). When s(k) is represented on the interval (-2048,2047), an RMSMIN of

$$\text{RMSMIN} = 50$$

produces a good tradeoff.

2.5.2 Adaptive Quantizer

The prediction error e(k) is the input to the adaptive quantizer whose basic design is illustrated in Fig. 2.4. The input is normalized by an adaptive scaling factor $\sigma(k)$ and the result is compared to a set of thresholds T_i. The recommended thresholds are symmetric and are illustrated in Fig. 2.5 and listed in Table 2.1. The level in which the normalized input falls specifies the quantizer output q(k). The inverse quantizer output $\hat{e}(k)$ is the quantized version of the quantizer input. It is the product of a scaling factor f(q(k)) and the state variable $\sigma(k)$. The recommended scale factors are tabulated in Table 2.2. The recommended thresholds were computed to be equidistant between the scaling factors.

The state variable $\sigma(k)$ is designed to be an approximation to the standard deviation of e(k). Most of the time the scaled average of $|\hat{v}(k)|$ is an acceptable estimate. However, in voiced speech at the beginning of a pitch period, e(k) is much larger than usual. Therefore, whenever one of the outermost quantizer level occurs, $\sigma(k)$ is significantly increased. If no further outer level occurs, $\sigma(k)$ decays back to the scaled average of $|\hat{v}(k)|$. Thus, $\sigma(k)$ is updated by

$$\sigma(k) = \max\{\text{SMIN} < |\hat{v}(k)| > , \phi[q(k)]\sigma(k-1)\} \qquad (2.13)$$

The first term in the braces of Eq. (2.13) usually dominates. This means that the quantizer behavior is largely determined by SMIN$< |\hat{v}(k)| >$ and, hence, by the product of SMIN and RMSMIN. It is recommended that the scale factor SMIN be set to 0.3.

The second term in the braces only affects performance at the beginning of pitch periods. The quantizer expansion factors $\phi[q(k)]$ are given in Table 2.2.

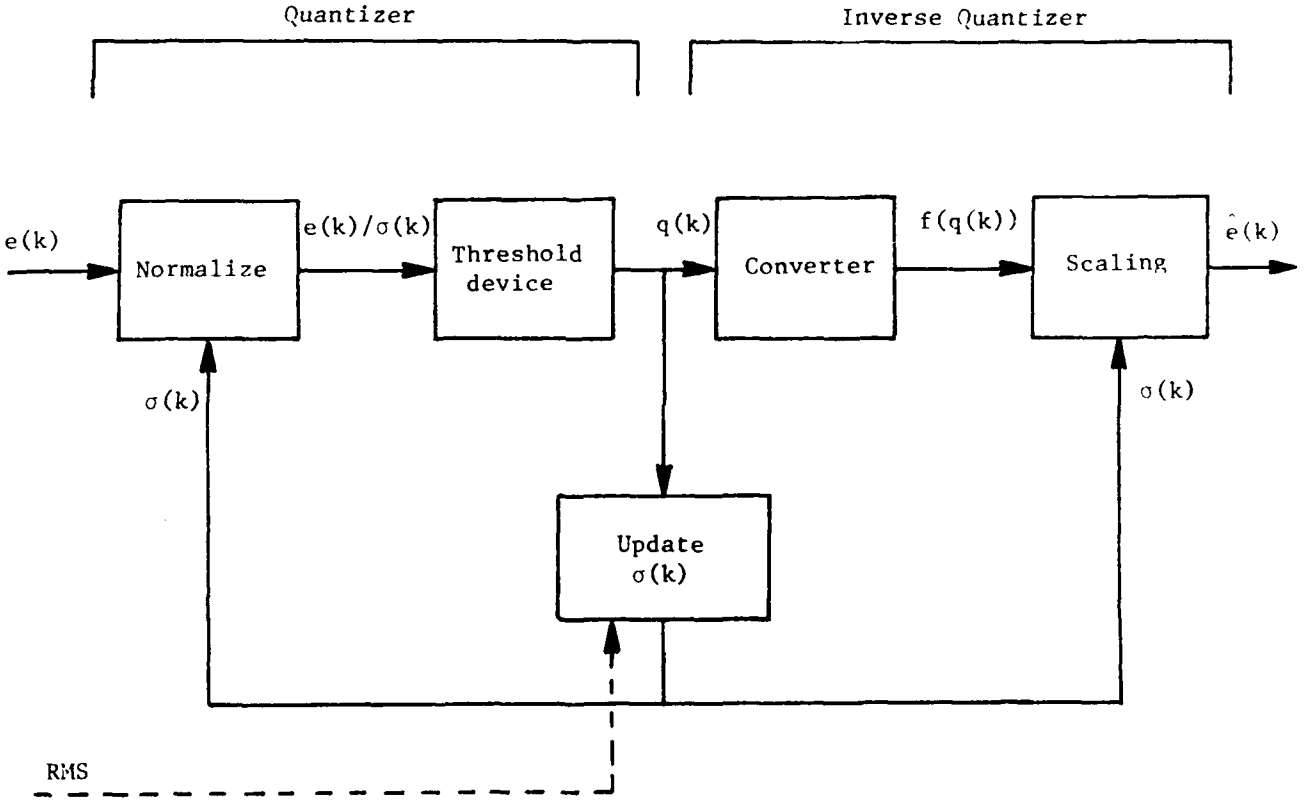


Fig. 2.4 Adaptive Quantizer

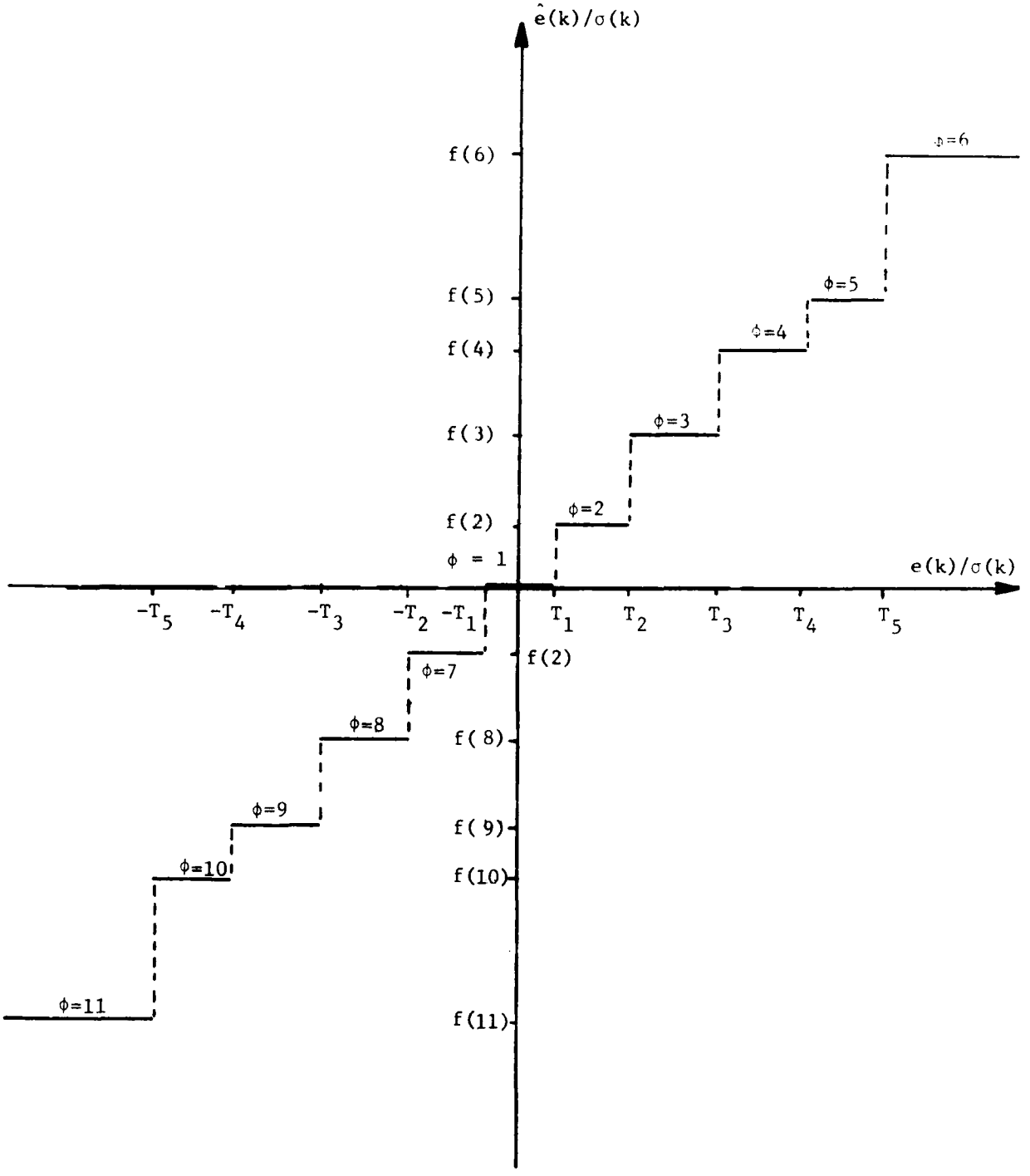


Fig. 2.5 Quantizer Thresholds and Output Levels

Table 2.1 Quantizer Thresholds

i	T_i
1	0.90
2	3.03
3	5.38
4	7.75
5	10

Table 2.2 Quantizer Scaling Factors and Expansion Factors

q(k)	f[q(k)]	ϕ[q(k)]
1	0.00	0.7
2,7	$\pm$ 1.80	0.8
3,8	$\pm$ 4.25	0.9
4,9	$\pm$ 6.50	1.1
5,10	$\pm$ 8.00	1.5
6,11	$\pm$ 12.00	2.2

2.6 Pitched Repetition

Even with adaptive low-pass filtering, sections of voiced speech can generate a large number of bits rapidly. This could cause the SAMPLE buffer to overflow. To avoid the overflow problem, the system uses a technique known as Pitched Repetition. When overflow is imminent, a block of samples is deleted at the transmitter and replaced at the receiver with previous reconstructed speech. The replacement samples are the reconstructed samples from the previous pitch period. In order to have an absolute control on the overflow condition, the repetition size has to be carefully set. During a voiced region, the bit rate may be as high as 3.4 bits per sample. Thus, the repetition size of 80 samples is enough to force more than 126 samples to be transmitted in a time slot, i.e., the total number of output samples is more than that of input samples. The decision to use pitched repetition is made at the beginning of each block and a special signal is used to alert the receiver.

If the SAMPLE buffer contains more than 850 samples at the beginning of a block-time, pitched repetition is employed. First, the pitch period T is computed in the usual way. The output pointer in the SAMPLE buffer is then moved forward 80 samples. Thus, these 80 samples will not be processed by PARC. They are, however, involved in the β calculation which takes place with the output pointer at its new location.

The receiver must still produce an output for those samples which are not processed by PARC. It does this by using previous outputs delayed by one pitch period. Thus, if the first sample skipped is $s_f(k)$, it is represented at the receiver by $\hat{s}(k-T)$. Similarly, $s_f(k+1)$ is represented by $\hat{s}(k+1-T)$ and so on. The transmitter must know this since it uses $\hat{s}(k)$ values in computing v(k). Therefore, the $\hat{s}(k)$ buffer is filled

with prior $\hat{s}(k-T)$ values as part of the pitched repetition. After this, PARC resumes normal operation.

This method of repeating short intervals of samples from the previous pitch period works with little distortion during voiced speech because samples one pitch period apart are highly correlated. In this way, the buffer overflow problem is overcome with a minimum of additional distortion.

The receiver must be informed that a period of pitched repetition is taking place. As detailed in the next section, this is accomplished through use of the unused values of β.

2.7 Noiseless Source Coder

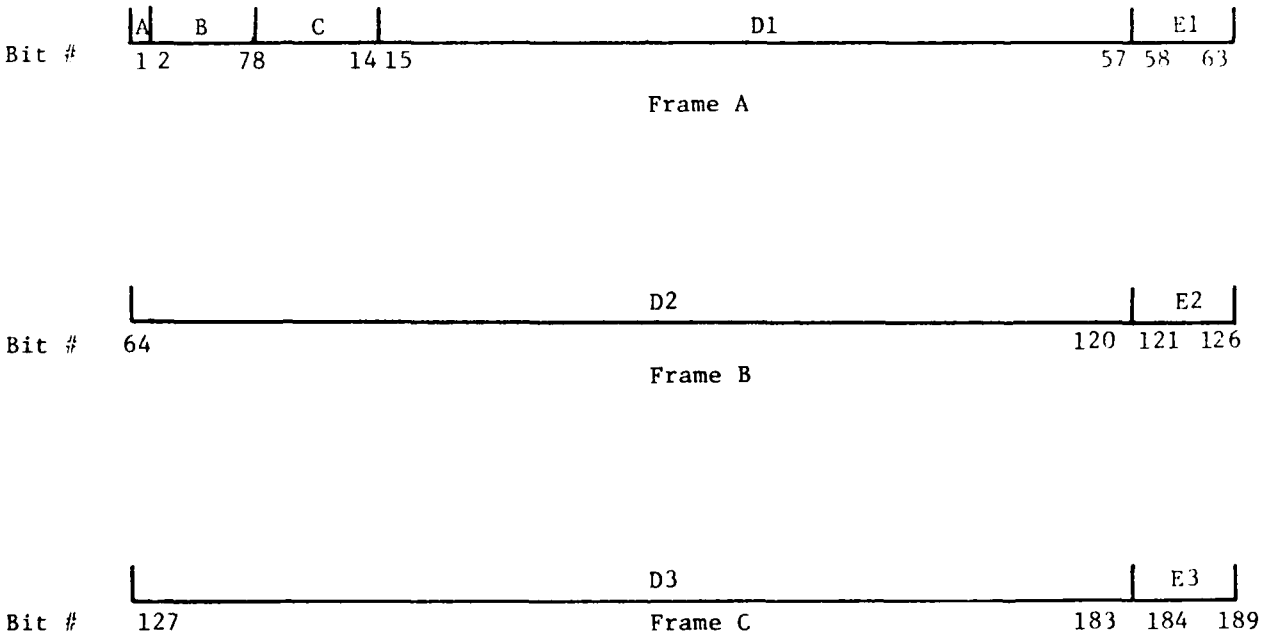
The function of the noiseless source coder is to combine all of the information from PARC and produce the corresponding bit stream for transmission. The output of the coder are blocks of 189 bits. Each block represents one set of values of β and T and N_B level variables. In addition, the block provides for synchronization and error control.

The format of a typical block is illustrated in Fig. 2.6. The block is divided into three 63-bit frames to facilitate error control. The last six bits of each frame, denoted E1, E2 and E3 in Fig. 2.6, are used for error correction. A single-error-correcting (57,63) Hamming code is used. The first 57 bits in each frame are available for information. The (57,63) codes can each correct any single error so up to three errors per block can be corrected.

The first bit in the block, shown as Field A, is for block synchronization. It is set to 0 in odd numbered blocks and 1 in even numbered blocks. The receiver detects this pattern and knows where the block begins.

The next field in the block is six bits long and contains information specifying on the pitch period T. The pitch period is constrained to be an integer between 20 and 83, so six bits can transmit the pitch period without quantization error.

Field C in a normal frame is seven bits long and contains information on the pitch correlation coefficient β. It was found that β can be quantized fairly coarsely with negligible degradation, so that 97 possible values in the range of [-2,2] are allowed. This means that 97 of the possible 128 values of Field C are used to represent β. If one of the other 31 patterns appears, it indicates that this is not a normal block. Rather, it is one using pitched repetition. The pitched repetition blocks are discussed later.



Field	Content
A	Synchronization Bit
B	Pitch Period T
C	Pitch correlation coefficient β
D1,D2,D3	Quantizer levels
E1,E2,E3	Parity bits

Fig. 2.6 Normal Block Format

Following the β field in a normal block are the 157 bits representing the quantizer levels. These are denoted as Fields D1, D2 and D3. The source code used for the quantizer levels are described in Table 2.3. Quantizer levels 2 through 11 are each represented by a variable length bit pattern. Quantizer level 1, however, occurs so often that there are two ways of representing it. Isolated occurrences are represented by the 1-bit sequence 0. If level 1 occurs 14 times in a row, the entire string is represented by the sequence 1110. Thus, the source code is an over-full variable-length to variable-length mapping.

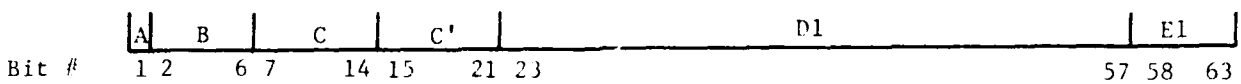
There is also a bit pattern associated with the null quantizer level sequence; this is used to fill out a block when the samples in the sample buffer do not generate at least 157 bits. Because of the variable number of bits used for different quantizer level sequences, an integral number of samples will not always generate exactly 157 bits. If there are more than 157 bits generated by a set of samples, the excess bits are the first bits transmitted in the next block's quantizer level field.

The normal format described above is used under most circumstances. If pitched repetition block is to be signaled to the receiver, however, the block format is changed slightly in the first frame of 63 bits, as shown in Fig. 2.7. Pitched repetition is signaled by using a special bit pattern, a "false β", in the field C which is usually reserved for β. The next 7 bits are then taken from the quantizer level field D1 to create Field C'which transmits the actual β. Thus, in such a block, only 150 bits are used for quantizer levels.

To protect against missing a "false β", or deciding one in error, the bit patterns for the β were carefully selected. They were designed so that neither situation can occur due to a single bit error. The "false β"

Table 2.3 Description of Quantizer Level Source Code

Quantizer Level Sequence	Bit Sequence
1	0
7	100
2	101
8	1100
3	1101
14 1's	1110
9	111100
4	111101
10	1111100
5	1111101
11	11111100
6	11111101
null	11111110



Field	Content
A	Synchronization bit
B	Pitch period T
C	False β
C'	Pitch correlation coefficient β
D	Quantizer levels
E1	Parity bits

Fig. 2.7 First Frame During Pitched Repetition

is represented by 0000000. The 7 patterns with a single one and the 21 patterns with two ones are never transmitted. Thus, only 99 patterns are available for true β values. At the receiver, if field C contains either all zeros or a single 1, it is interpreted as the "false β". If no more than one channel error has occurred, this will happen if and only if a "false β" was actually transmitted.

2.8 Receiver

The principal elements of the receiver are illustrated in Fig. 2.8. The received bit sequence b(m) is monitored by a synchronizer which establishes the beginning of a block. The decoder transforms the bits in a block into the quantizers levels q(k) and the pitch parameters β and T. These are then processed by the inverse of the PARC algorithm. The $\hat{s}(k)$ values are stored in a variable delay receiver buffer. They are clocked out of the buffer and interpolated to produce the un-sampled output signal $\hat{s}_{out}$. The primes on all of these quantities introduced earlier to account for possible channel errors have been dropped for notational simplicity.

The synchronization actually operates in two modes: initial establishment and monitoring. During the establishment phase, the rest of the receiver system is disabled. The synchronizer looks for a sequence of bits spaced by 189 bit-times whose polarity oscillates. When a sequence of 10 bits with perfect oscillation is found, the synchronizer decides that it must represent the sync bit in 10 successive 189-bit blocks. The rest of the receiver is enabled and the synchronizer changes to the monitor mode. If it detects a significant number of errors in the sync bits, it assumes block synchronization has been lost. The receiver is disabled and the synchronizer returns to the synchronization mode.

Once synchronization has been established, the decoder can go to work. It basically inverts the operations performed by the noiseless source coder. Thus, it works on a full block of bits. The Hamming codes are decoded and any correctable bit errors are corrected. The values of β and T are found and the sequence of quantizer level values is formed.

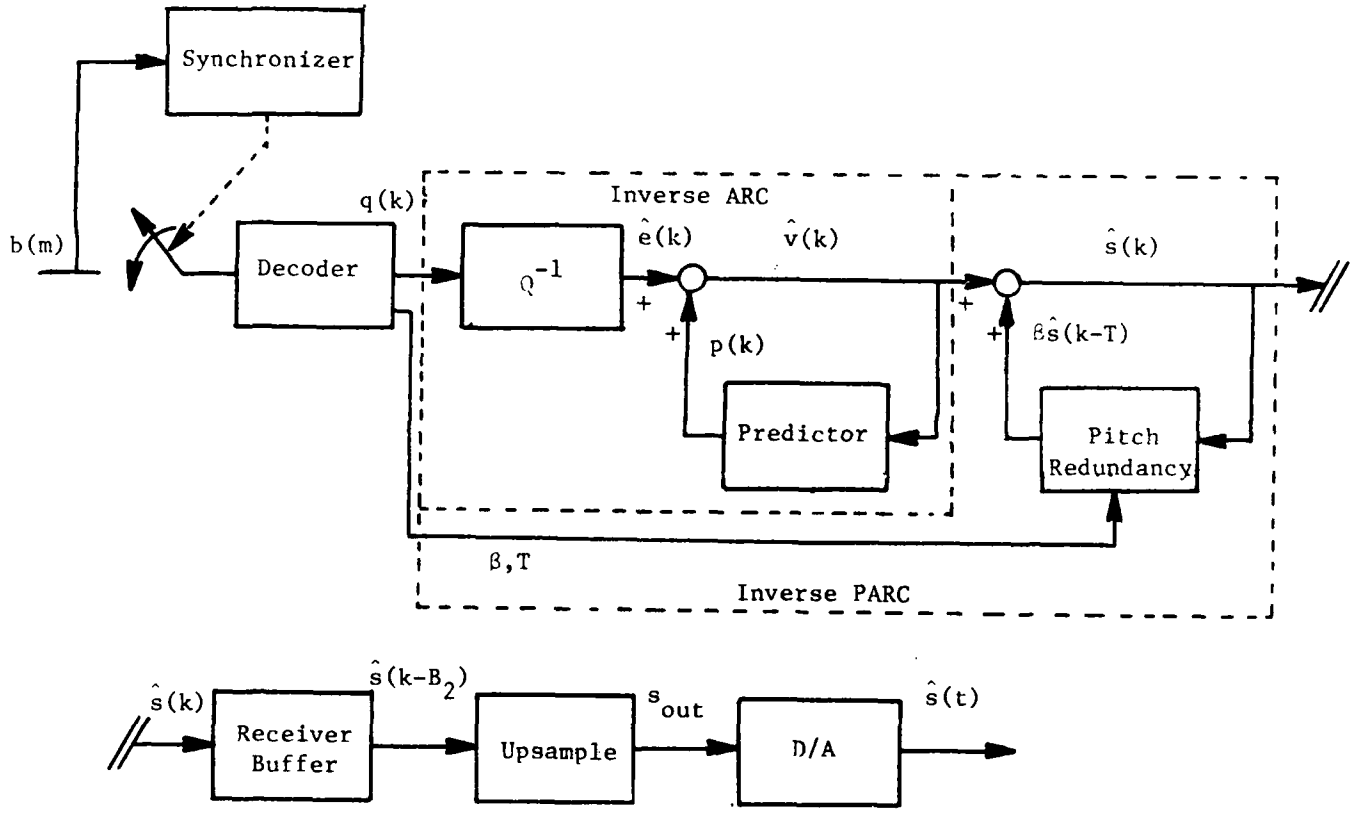


Fig. 2.8 Receiver Structure

If pitched repetition is being used, a special code is set. All of this information is placed in a double-buffer which interfaces with the PARC receiver.

The PARC receiver processes a full block of data. The S and T values are fixed for the block but the number of samples handled is the variable N_B. If pitched repetition is called for, the old $\hat{s}(k)$ values are produced before the ARC receiver is enabled.

The output $\hat{s}(k)$ from the PARC receiver are stored in the circular receiver buffer. This buffer complements the variable delay transmitter buffer and requires a carefully designed control algorithm. If there are no channel errors, the total delay for the system will be a constant. Therefore, the sum of the transmitter delay B_1 and the receiver delay B_2 will be fixed.

The buffer control logic in the decoder is designed to prevent the received sample buffer from ever overflowing or underflowing. In normal operation neither of those can happen but channel errors can add or delete samples. Since samples are removed from the buffer at a known rate, the buffer control logic will always know how many samples are in the buffer. If there is not enough room in the receiver buffer for all of the $\hat{s}(k)$ in a block, the excess $q(k)$ are simply discarded. Since a full receive buffer corresponds to an empty transmitter buffer, this usually occurs during silence. The deletion of silence is generally not a problem.

As the transmitter buffer fills, the receiver buffer empties. If channel errors have caused the deletion of samples, it is possible for the receiver to run out of $\hat{s}(k)$ values. This underflow condition is also prevented by the buffer control logic. If the number of $\hat{s}(k)$ values in a block plus the number of $\hat{s}(k)$ stored in the buffer do not total at least 126,

additional q(k) representing silence are added to the LEVEL buffer. Thus, there are always enough.

The upsampling has been added to reduce the sampling noise caused by the non-ideal nature of the analog output filter. The 6400 samp/sec sampling rate is at the Nyquist rate of the 3200 Hz output filter. Thus, severe aliasing is possible. The upsampler effectively increases the sampling rate to 12800 samp/sec. It does this by interpolating between successive $\hat{s}(k)$ values. It was found that linear interpolation was sufficiently accurate to greatly reduce the aliasing.

CHAPTER 3
SYNCHRONIZATION

Although PARC is basically a sequential algorithm, the use of pitch redundancy reduction and error control forces on it a block structure. The transmitter quantizes a block of N_B speech samples using a set of pitched reduction parameters and encodes this into frames of binary information. At the other end, the receiver must identify these frames to be able to properly decode the information it receives. This necessitates some sort of synchronization between the transmitter and the receiver. In this chapter, the synchronization technique used in PARC and its implementation is described. Further, the operation is analyzed to illustrate its satisfactory performance.

There are two aspects to the synchronization operation performed in the receiver. The receiver must first locate the frame boundaries in the received bit stream. This is called synchronization acquisition. After acquisition, it must monitor the frame boundaries on a continuing basis to ensure that sync is not lost. This is called synchronization monitor. Contract requirements specify that bits are not dropped during transmission. Therefore, once synchronization is properly acquired, there should be no way of losing it. However, there are several reasons for providing the sync monitor. Sync acquisition is a probabilistic operation; and although there is a high probability of acquiring sync properly in one attempt, in case of a wrong decision, the sync monitor provides a way for re-attempting sync acquisition. There are other abnormal conditions that inevitably occur during algorithm development and testing which can also cause sync to be lost. Some of these are bad connections on the digital I/O connectors, faulty

IOS operation (see Chapter 12), reinitialization of the algorithm at one end of the communication system. All these make it imperative to provide the algorithm with the sync monitor, in other words, with re-syncing capabilities to ensure proper uninterrupted operation.

3.1 Sync Algorithm Description

The technique selected here to synchronize the receiver and the transmitter is similar to that used in the T1 Carrier System. A bit pattern called the synchronization pattern is selected. One bit at a time from the predetermined pattern is interjected at regular intervals into the bit stream generated at the transmitter. In this system, the sync bit is inserted at the beginning of each frame of 189 bits generated by processing a block of N_B samples. The receiver tries to locate the sync pattern embedded in the received bit stream, thereby locating the frame boundaries.

Any bit pattern can be used for the sync pattern as long as it does not coincide with some naturally generated pattern at the transmitter. Using a shorter sync pattern reduces the memory space and computation required at the receiver during sync acquisition. After some consideration, a two bit pattern 01 was selected for the synchronization pattern.

The following subsections detail the algorithm for the two aspects of synchronization. A short analysis is also presented with each to get some idea of the performance of these operations.

3.1.1 Synchronization Acquisition

There are two considerations in selecting this algorithm. First, it should not take too long for each acquisition operation. And secondly, the probability of making the right decision should be reasonably high to ensure that the right synchronization is achieved in a couple of attempts, if not in one.

The sync acquisition algorithm consists of segmenting the received bit stream into blocks of 189 bits each. One of the 189 bits is the sync bit,

and the corresponding bit position follows the sync pattern over the blocks of received bits. The decoder generates the sync pattern at the receiver, and checks its correlation with each of the 189 bit positions. The position that correlates exactly with the sync pattern for 10 blocks is picked to be the sync bit, marking the beginning of subsequent frames.

For sync acquisition to be unambiguously successful, only the sync bit of the 189 bits in a frame must correlate exactly with the sync pattern for 10 blocks. If there were no transmission errors, the problem here would consist of picking a deterministic sequence from the midst of a stochastic process. However, the received sync pattern is corrupted by transmission errors, an average error rate of 1%. Each of the other 188 bits are random 1's and 0's, and they correlate with the sync pattern with a probability of 0.5. With these and the assumption that the channel affects the bits independently, the probability of making a correct decision about sync acquisition can be determined.

The probability that the sync bit is transmitted without errors for 10 consecutive blocks is

$$\alpha = (0.99)^{10} = 0.9044$$

The probability that one of the other bit positions correlates perfectly with the sync pattern for 10 consecutive blocks is

$$\beta = (0.5)^{10} = 0.000976$$

The probability that none of the other 188 bit positions correlates perfectly with the sync pattern is $(1-\beta)^{188}$. The probability of an unambiguous sync acquisition decision is

$$\alpha(1-\beta)^{188} = 0.9044 \times 0.8322 = 0.7526$$

Thus, the probability of successful sync acquisition in several attempts can be computed:

In one attempt, P{sync acquisition} = 0.7526, P{failure} = 0.2474

In two attempts, P{sync acquisition} = 0.9388, P{failure} = 0.0612

In three attempts, P{sync acquisition} = 0.9848, P{failure} = 0.0151

The implementation of this algorithm is slightly different from the description here. This was done to reduce the amount of computation required.

3.1.2 Synchronization Monitor

There are three considerations in selecting the algorithm here. The sync monitor checks to see if the received sync bits follow the sync pattern. Because of transmission errors, there are some errors in the received sync bits inspite of the error control. After allowing for these errors, the algorithm should decide that sync is retained. The probability of erroneously deciding that sync is lost should be extremely small to ensure that the receiver operates uninterrupted for long periods of time. Secondly, if sync is lost, the algorithm should realize this in a reasonably short time. And finally, the algorithm should be computationally simple to implement.

With these considerations in mind, the following algorithm is suggested. A correlation between the received sync bit S_i and the expected sync bit $\hat{S}i$ is computed. Based on the correlation, a value is assigned to a r.v. x_i.

$$x_i = \begin{cases} +2 & \text{if } S_i \neq \hat{S}_i \\ -1 & \text{if } S_i = \hat{S}_i \end{cases}$$

This variable is used to update a sync variable v_i.

$$v_i = v_{i-1} + x_i$$

If at any time, the variable v_i exceeds the threshold T, it is decided that sync has been lost. The variable v_i starts with an initial value 0, and is constrained to be non-negative for all i. If its value drops below 0, it is reinitialized to 0. The threshold T used here is 12.

This algorithm has the effect of switching the rate of change of the sync variable from +2 when $(S_i \neq \hat{S}_i)$ to -1 when $(S_i = \hat{S}_i)$. If the channel error rate r is less than 1/3, the rate of change is negative. The sync variable decays to 0 and stays there. If the channel error rate is greater than 1/3,

the rate of change is positive and the sync variable drifts towards the threshold T. The maximum positive slope is 2 and occurs when the channel error rate is 1.

The rate of change x_i is a binary random variable. Over a period of time, its average is the slope Δv of the sync variable. The slope Δv can be used to determine the time before sync is lost.

$$t = T/\Delta v$$

The slope Δv is a binomial random variable. Its function t, the time before sync is lost, which is also a random variable can be described by a distribution similar to the binomial distribution.

$$P(t) = \frac{\Gamma(t + 1)}{\Gamma(t/3 + 5)\ \Gamma(2t/3 - 3)}\ r^{(t/3 + 4)}\ (1 - r)^{(2t/3 - 4)},\ t \geq 6$$

Its expected value is

$$\langle t \rangle = T/(3r - 1)$$

Using these, some estimates of the performance of the sync monitor algorithm can be obtained. While proper sync is retained, the error rate for the sync bit is reduced to 0.005 by the error control. At this error rate, the algorithm would never lose sync. If the error rate deviated from its average value to 1, it would take 6 frames for sync to be lost. The probability of this can be computed to be 1.5625×10^{-14}.

If sync is actually lost, the average error rate for the sync bit is 0.5. The algorithm would take an average of 20 frames before deciding it has lost sync.

The implementations of these two algorithms are described in Chapter 12. The sync acquisition algorithm is implemented slightly differently from its

description here to reduce the computation involved. The sync monitor algorithm is implemented as described here using simple logical and shift operations.

CHAPTER 4

PITCH EXTRACTION STUDIES

4.1 Introduction

The aim of efficient coding methods is to reduce the channel capacity required to transmit a signal with specified fidelity. To achieve this objective, it is desirable to reduce the redundancy of the transmitted signal. One well-known procedure for removing signal redundancy is predictive coding. In predictive coding, redundancy is removed by subtracting from the signal that part which can be predicted from its past. The PARC system is essentially APC system which includes a pitch extraction loop for long-term redundancy removal.

In this chapter, several studies concerning pitch redundancy removal in PARC are described. The correlation technique, as well as AMDF algorithm, for pitch extraction is outlined in Section 4.2. The complete algorithm with pitch extraction loop was simulated on a digital computer; simulation results are discussed in Section 4.3.

4.2 Pitch Extraction Algorithms

It is well known that voiced speech is highly correlated from pitch period to pitch period [1]. The long term prediction of s(k) is given by

$$\hat{s}(k|k-T) = \beta s(k-T) \tag{4.1}$$

Here β is a scalar which depends on the correlation between s(k) and s(k-T) while T is an estimate of the pitch period (in samples). The use of β reflects the amplitude changes of speech signal which occur from period to period especially during the beginning and end of the voiced segments. For unvoiced speech, β is generally small and long-term prediction is relatively ineffective. The long-term prediction $\hat{s}(k|k-T)$ is subtracted from s(k) to form the pitch-reduced-speech $v(k) = s(k) - \hat{s}(k|k - T)$.

The goal in selecting β and T is to minimize the error

$$E_1 = \frac{1}{K} \sum_{j=1}^{K} [s(j)-\beta s(j-T)]^2 \tag{4.2}$$

Here block adaptation with block length of K has been assumed. The choice of K depends on various factors and will be discussed in a next section. The derivative of E_1 with respect to β yields

$$\frac{\partial E_1}{\partial \beta} = \frac{2}{K} \sum_{j=1}^{K} [s(j) - \beta s(j-T)]\, s(j-T) \tag{4.3}$$

Equating this derivative to zero and solving for β gives

$$\beta = \frac{\sum_{j=1}^{K} s(j)s(j-T)}{\sum_{j=1}^{K} s^2(j-T)} \tag{4.4}$$

If this result is substituted in Eq. (4.2), the equation becomes function of T alone given by

$$E_1(T) = \frac{1}{K}\sum_{j=1}^{K} s^2(j) - \frac{\frac{1}{K}\sum_{j=1}^{K} s(j)s(j-T)}{\sum_{j=1}^{K} s^2(j-T)} \qquad (4.5)$$

Therefore to minimize E_1 with respect to T it is necessary to maximize the rightmost term of equation (4.5). The approach used was to compute

$$A(T) = \frac{\sum_{j=1}^{K} s(j)s(j-T)}{\sum_{j=1}^{K} s^2(j-T)} \qquad (4.6)$$

for all values of T, $T_{min} \leq T \leq T_{max}$ and then select the value for which A(T) is maximum. The lower limit, T_{min}, of the search range was selected to be smaller than minimum value of pitch periods for different speakers while the upper limit, T_{max}, is influenced by various factors such as number of bits available for transmission of T's, processing time limitations due to real time application and maximizing energy reduction.

The above method, though simple to implement, involves extensive computation. For example, if the block length is K and searching range is R then for each value of R there are 2K+2 multiplications and K additions. Hence the total number of multiplications for finding the pitch period becomes R(2K+2); if R=100 and K=100 then this number is 20200 (this is just for one block of 100 samples. This many multiplications consume significant processing time which is crucial in real time implementation in Macro Arithmetic Processor (MAP).

The stringent requirement on timing in the MAP, led to a modification of the above correlation technique for pitch extraction. This is done by forming Average Magnitude Difference Function (AMDF)[2]

$$A'(T) = \sum_{j=1}^{K} |s(j) - s(j-T)| \qquad (4.7)$$

$$T_{min} \leq T \leq T_{max}$$

It is easy to see that for any periodic function, the above sum is minimum for T equal to the period. Hence, the pitch parameter T was determined by minimizing the function A'(T) with respect to T. The gain parameter was obtained by substituting this value of T into Eq. (4.4). The computational saving in this method is apparent since there is no multiplication involved.

This modification of correlation method gives exactly the same values of T (and hence of β) in voiced speech but differs in unvoiced speech. However, unvoiced speech is non-periodic and T is arbitrary and hence not important.

Figure 4.1 shows the plot of β and T against speech samples. The correlation coefficient β jumps to a high value for a voiced speech block which is followed by silence. This high value amplifies the quantization noise thus decreasing overall signal to noise ratio. Limiting β to [-2,2] was found to eliminate this problem and give satisfactory performance. Further discussion of limiting incontext with quantization of β and coding is presented in Chapter 11.

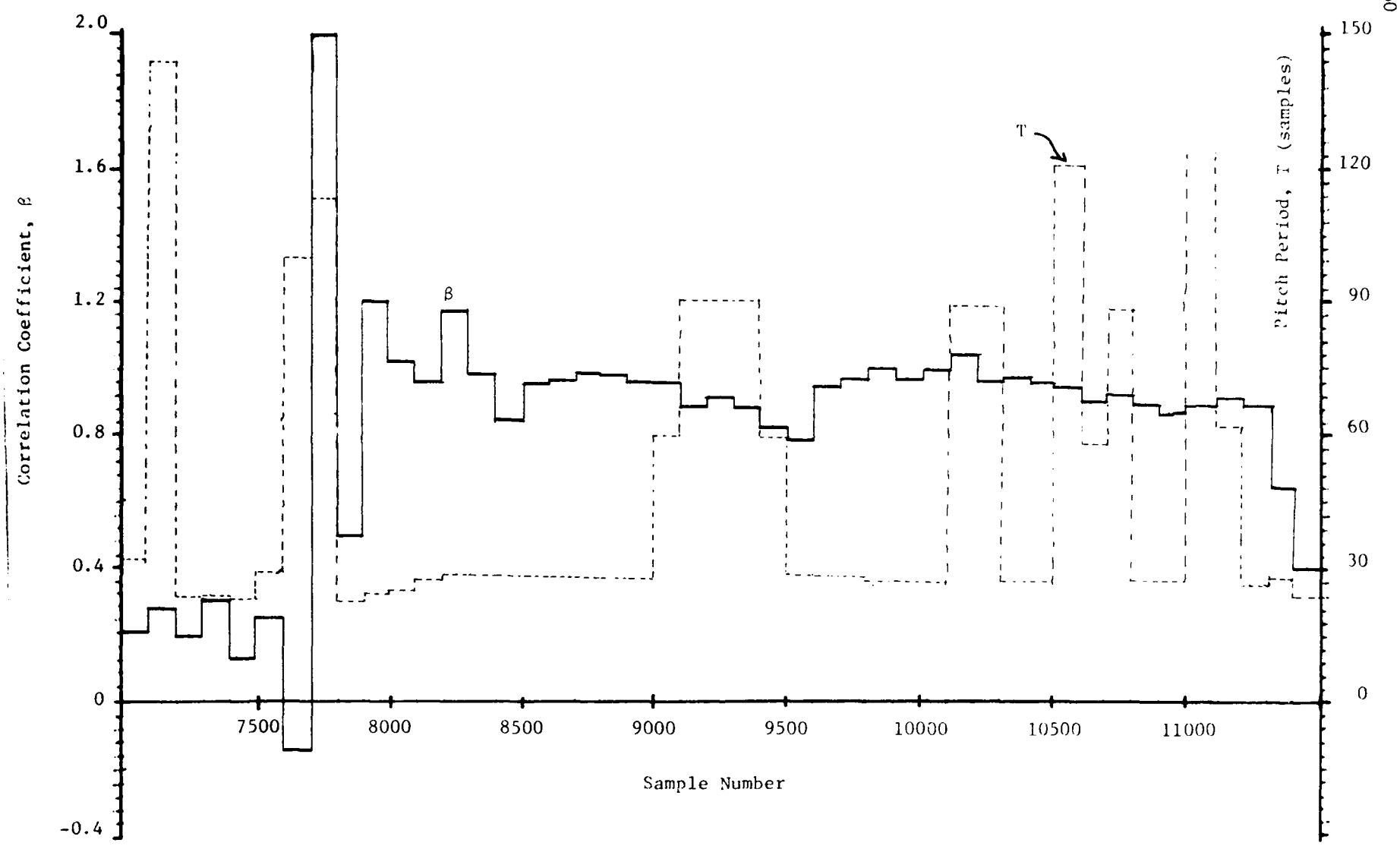


Fig. 4.1 Pitch Period T and Correlation Coefficient β vs Sample Number. (Sent 1: Male Speaker)

PARC algorithm was simulated on PDP 11/60 computer. The following phonetically balanced sentences were chosen for the simulation work.

Male speaker: sent 1 -"Cats and Dogs each hate the other."

Female speaker: sent 11 -"The pipe began to rust while new."

Beta's and T's were extracted using both the correlation and AMDF methods discussed above. The results of this study are shown in Table 4.1.

TABLE 4.1a

Comparison of Correlation and AMDF Method
for Pitch Extraction

Sentence 11: Female Speaker

Sample No.	Correlation Technique		AMDF Algorithm	
	β	T	β	T
3500	-0.26	23	0.06	50
3600	-0.23	25	2.00	134
3700	0.42	22	0.57	147
3800	0.85	58	0.85	58
3900	1.03	29	1.03	29
4000	0.95	29	0.95	29
4100	0.91	88	0.91	88
4200	0.93	88	0.93	88
4300	0.99	29	0.99	29
4400	0.99	29	0.99	29
4500	0.80	29	0.80	29
4600	1.02	29	1.02	29
4700	1.00	29	1.00	29
4800	0.92	29	0.92	29
4900	0.61	28	0.61	28
5000	0.31	25	0.24	89
5100	0.32	20	0.32	20
5200	0.56	94	0.41	120
5300	0.30	62	0.24	23
5400	0.93	27	0.93	27
5500	0.87	29	0.87	29
5600	0.89	30	0.89	30
5700	0.99	30	0.99	30
5800	1.01	30	1.01	30
5900	1.01	30	1.01	30
6000	1.07	30	1.07	30
6100	1.07	30	1.07	30
6200	1.08	30	1.08	30
6300	1.04	30	1.04	30
6400	1.01	30	1.01	30
6500	0.97	30	0.97	30
6600	0.94	30	0.94	30
6700	0.86	30	0.86	30
6800	0.79	30	0.79	30
6900	0.66	31	0.66	31
7000	0.19	63	0.04	27

TABLE 4.1b

Comparison of Correlation and AMDF Method
for Pitch Extraction

Sentence 1: Male Speaker

Sample No.	Correlation Technique		Sample No.	AMDF Algorithm	
	β	T		β	T
1700	0.53	141	1700	0.37	38
1800	0.35	141	1800	0.35	141
1900	0.48	135	1900	0.32	22
2000	0.11	148	2000	-0.25	133
2100	0.30	27	2100	0.20	145
2200	1.45	42	2200	1.45	42
2300	0.71	48	2300	0.71	48
2400	0.76	50	2400	0.76	50
2500	0.93	52	2500	0.93	52
2600	0.74	53	2600	0.74	53
2700	0.96	53	2700	0.96	53
2800	0.94	53	2800	0.94	53
2900	0.94	54	2900	0.94	54
3000	0.94	55	3000	0.94	55
3100	0.90	57	3100	0.90	57
3200	0.72	59	3200	0.72	59
3300	0.60	61	3300	0.60	61
3400	0.54	62	3400	0.52	63
3500	0.79	24	3500	-0.98	31
3600	1.16	52	3600	1.16	52
3700	1.21	53	3700	1.21	53
3800	1.01	53	3800	1.01	53
3900	0.89	53	3900	0.89	53
4000	0.81	55	4000	0.81	55
4100	0.97	57	4100	0.97	57
4200	0.98	58	4200	0.98	58
4300	1.00	58	4300	1.00	58
4400	0.96	59	4400	0.96	59
4500	0.85	59	4500	0.86	60
4600	0.92	60	4600	0.92	60
4700	0.84	61	4700	0.84	61
4800	0.74	62	4800	0.75	63
4900	0.69	65	4900	0.69	65
5000	0.60	66	5000	0.60	66
5100	0.60	68	5100	0.61	67
5200	0.42	65	5200	0.44	67
5300	0.93	64	5300	0.93	64
5400	0.71	64	5400	0.71	64
5500	0.87	65	5500	0.88	66
5600	0.66	131	5600	0.66	131
5700	0.33	66	5700	0.34	67
5800	0.32	62	5800	0.32	62

4.3 Redundancy Removal

As described earlier, the APC system is based on the removal of two kinds of redundancy: short term redundancy caused by vocal tract filter and long-term redundancy caused by pitch frequency. Once the pitch period T and gain parameter β are determined, reduced speech is formed as

$$v(k) = s(k) - \beta \hat{s}(k|k-T) \tag{4.8}$$

where $\hat{s}(k|k-T)$ represents the reconstructed speech sample. Figure 4.2 shows the plot of reduced and original speech. It is easy to notice the energy reduction achieved in the almost periodic voiced portion of the speech. The amount of energy reduction achieved is expressed by SER (Signal Energy Reduction) which is calculated as

$$SER = -10 \log_{10} \frac{\sum v^2(k)}{\sum s^2(k)} \tag{4.9}$$

As the value of signal energy reduction is increased the dynamic range of the input signal to quantizer is reduced; hence the reduced speech signal can be represented by the lower quantizer levels thus requiring fewer bits for transmission. The SER can be increased by accurately picking pitch period T and choosing the block size such that effect of transition of β from block to block is minimum. The parameters β and T are associated with every block. For smaller block sizes, the number of parameters to be transmitted per second is increased. However, for smaller block size, the amplitude variations are closely represented by β and transition of β from block to block is smooth. These factors contribute to improve SER.

Table 4.2 shows the effect of block-size variation on SER. The following performance measures were also computed:

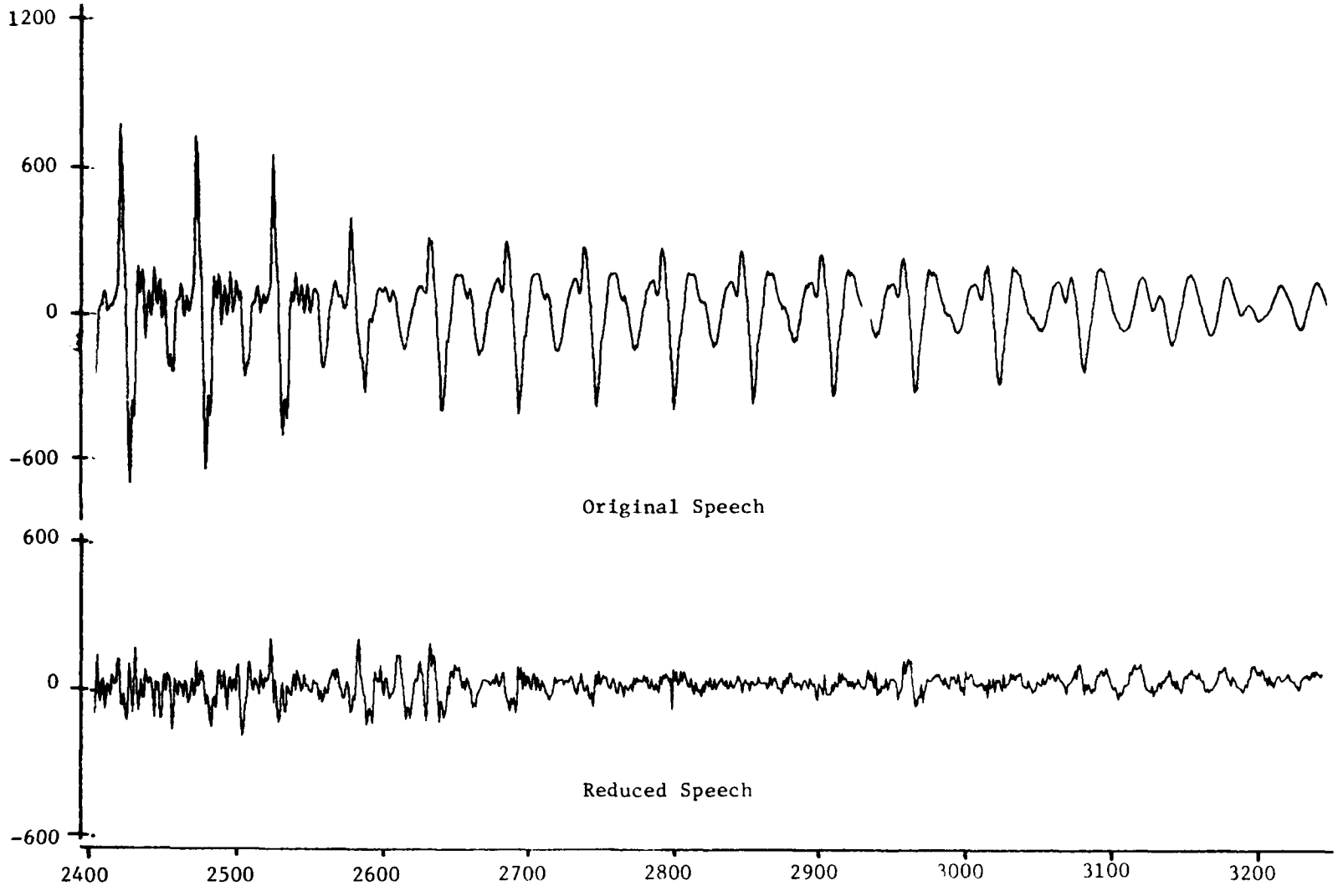


Fig. 4.2a Sentence 1, Male Speaker

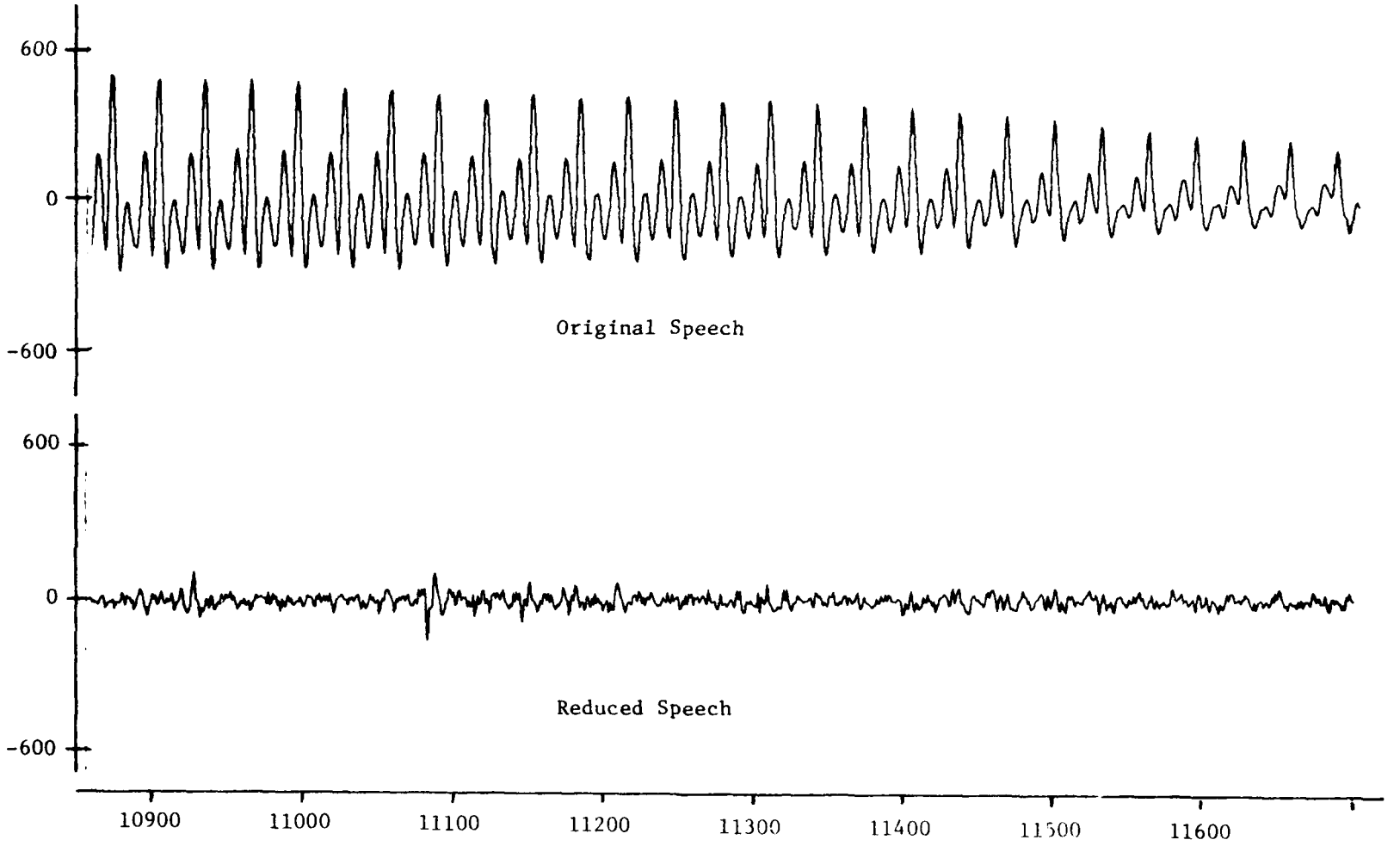


Fig. 4.2b Sentence 11, Female Speaker

Table 4.2

Effect of Block Size on Various Performance Measures

Sentence	Block Size	SER	Overall SNR	Inloop SNR	Entropy bits/sample
1	20	8.47 db	20.81 db	12.34 db	1.53
	40	7.07 db	20.24 db	13.17 db	1.48
	80	5.63 db	19.08 db	13.45 db	1.46
	100	5.44 db	18.97 db	13.53 db	1.43
	120	5.15 db	18.85 db	13.70 db	1.43
	140	4.70 db	18.26 db	13.56 db	1.44
	160	4.56 db	18.33 db	13.77 db	1.45
	180	4.41 db	18.31 db	13.90 db	1.42
	200	4.07 db	18.20 db	14.13 db	1.43
11	20	11.19 db	22.78 db	11.59 db	1.48
	40	9.81 db	22.44 db	12.63 db	1.44
	100	8.48 db	21.72 db	13.24 db	1.43
	120	8.04 db	21.65 db	13.61 db	1.42
	140	8.20 db	21.57 db	13.37 db	1.42
	160	7.92 db	21.59 db	13.67 db	1.43
	180	8.16 db	21.64 db	13.48 db	1.43
	200	8.33 db	21.58 db	13.25 db	1.44

$$\text{Overall SNR} = 10 \log_{10} \frac{\sum s^2(k)}{\sum [s(k) - \hat{s}(k)]^2} \tag{4.10}$$

$$\text{Inloop SNR} = 10 \log_{10} \frac{\sum v^2(k)}{\sum [v(k) - \hat{v}(k)]^2} \tag{4.11}$$

where $v(k) = \hat{v}(k) + n_q(k)$

$$\text{Entropy } H = - \sum_{i=1}^{11} p_i \log_2 p_i \tag{4.12}$$

where p_i = Probability of occurrence of i^{th} quantizer level.

As SER improves overall performance also improves. It is interesting to note that the SNR (overall) may increase even if the SNR (inloop) decreases because of the improvement in SER. The speech signal spectrum becomes flatter because of pitch extraction thus adversely affecting the performance of predictor in INLOOP. However, because of smaller dynamic range of reduced speech, the quantizer noise is decreased which more than compensates for the poor performance of predictor. Hence the overall performance improves.

The searching range ($T_{max} - T_{min}$) for pitch extraction also affects the redundancy removal. It was observed that a longer searching range gives better SER while a small value of searching range decreases SER by as much as 2 db. A searching range of the order of twice the maximum pitch period appears to be sufficient However, the longer searching range also means more computations and hence more CPU time. In the real-time simulations on the MAP, timing is critical and therefore the number of computations and memory transfers need to be reduced. In such cases the searching range must be reduced to achieve a compromise between the number of computations

and the reduction in SER that can be tolerated.

It was noticed that the pitch extraction algorithm sometimes picks double or triple pitch periods. This fact has only a modest effect redundancy removal. However, the transmission of double or triple pitch period values may require the allocation of more bits for transmission of T. Again, it is desirable to limit the search range.

In Section 4.2, the estimation algorithm used to compute the pitch period T and long term gain is based solely on the original speech. In fact, as seen by examining Fig. 2.10 the long-term redundancy removal operation actually subtracts the reconstructed speech from original speech. In an attempt to compensate for this fact, the β obtained from Eq. (4.4) was modified by multiplying by scalar α as

$$\beta^{*} = \alpha\ \beta \tag{4.13}$$

Here α can be expressed [3] as

$$\alpha = \frac{1}{1 + \alpha} \tag{4.14}$$

where α is the inverse of signal to noise ratio. The parameter α was varied between 0 and 1.2 with no significant improvement was noticed. See Table 4.3.

TABLE 4.3

The Effect of α on SER and Overall SNR

α	SER	Overall SNR
0	0 db	15.61 db
0.9	5.45 db	19.23 db
0.94	5.55 db	19.29 db
0.96	5.57 db	19.24 db
0.98	5.55 db	19.40 db
0.995	5.52 db	19.28 db
1.0	5.55 db	19.31 db
1.04	5.48 db	19.16 db
1.1	5.37 db	19.21 db
1.2	5.07 db	18.79 db

4.4 References

[1] Flanagan, J. L.: Speech Analysis, Synthesis and Perception, 2nd Ed., Springer-Verlag, N.Y., 1972.

[2] Ross, M., et al.: "Average Magnitude Difference Function Pitch Extractor," IEEE Trans. on Acoustics, Speech and Signal Processing, Vol. ASSP-22, pp 353-362, Oct. 1974.

[3] "Study of Sequential Estimation Methods for Speech Digitization", Department of Electrical Engineering, University of Notre Dame, Notre Dame, Indiana, June 1975.

CHAPTER 5
TREE CODING

5.1 Introduction

The concept known as tree coding was investigated as part of this study to evaluate its ability to improve performance. One particular algorithm, the (M,L) algorithm, was the basis for most of the investigation. A modified version of the (M,L) algorithm called adaptive tree searching was developed and investigated. Simulations indicated that adaptive tree searching marginally improved the performance of the PARC algorithm.

5.2 The (M,L) Algorithm

5.2.1 Description

The (M,L) algorithm is one of a number of algorithms which perform what is known as tree coding. The main idea of tree coding is to defer making a decision, in this case, which quantizer level should be used for a given sample, until a later time when it can be made in light of that which follows. Tree coding is useful in predictive or backward adaptive quantization systems, because the selection of a quantizer level affects the selection of quantizer levels in the future. This effect can be represented graphically in the form of a tree, where a node represents the "state" of the system as a result of selecting the sequence of quantizer levels leading to that "state", with a branch connecting the node to the node representing the previous "state". The tree is rooted by an arbitrary "state" at an arbitrary time, and evolves in time, with a new level of nodes added at each sample time.

The (M,L) algorithm operates in the following way, and is illustrated in Figure 5.1. At a given time, let us say that there are n nodes in the outermost level of the tree, and a new sample is to be quantized. A new level of nodes are then "grown" from the outermost level, representing all of the new possible "states". Thus, if there are k quantizer levels, there will be kn nodes in the new level. These new nodes are then ranked by some performance criterion, such as quantization noise. Nodes are next pruned from the tree. This is a key step, because it is this that prevents the task from growing exponentially. In order to insure that a quantizer level is selected for a sample in a finite amount of time, a

quantizer level decision is forced, if necessary, for the sample L time units ago, by picking the predecessor node, in the level L-levels in, of the best of the new nodes. All branches which do not stem from that predecessor are pruned. If more than M new nodes remain, the M best nodes are kept, and the rest are pruned. The process then starts over.

5.2.2 Results

The (M,L) algorithm has been studied by a number of investigators in connection with simpler quantization schemes, such as adaptive delta modulation (ADM) and adaptive differential pulse code modulation (ADPCM). For example, Jayant and Christensen reported a 3dB improvement in the signal-to-noise ratio (SNR) using the (M,L) algorithm with simple ADM and ADPCM schemes, with M=4 and L=7. In contrast, our investigation concentrated on the use of the (M,L) algorithm in connection with adaptive residual coding (ARC). It was originally envisioned that the resulting algorithm would be embedded within a pitch extraction loop. Some preliminary work was done, however, without the pitch extraction loop in order to verify the tree coding software and to gain some insight into its operation. For example, by using a four level fixed quantizer and a third order fixed predictor in the ARC algorithm, an improvement of 3dB in the SNR was achieved for M=4 and L=7. Unfortunately, the results obtained using the standard ARC algorithm did not show as great an improvement in SNR.

In order to make the following results more understandable though, a comment is necessary here. One of the key elements of the ARC algorithm is the source coding, which translates quantizer levels into bit patterns. The selection of a good source code is dependent, however, on the statistics of that which is to be encoded. As a result, the performance of the (M,L) algorithm was evaluated without use of a source coder. Instead, the entropy of the quantizer levels was computed, which allows for a fair comparison between the possibilities. In general, a higher entropy allows better performance.

Some of the first results obtained were for the first second of Sentence

1, "Cats and dogs each hate each other". In this series of runs, a five level quantizer was used in the standard ARC algorithm, M was set to five, and L was varied. The results are shown in Table 5.1. These results illustrate the typical effect of tree coding: increasing L improves performance by increasing the SNR and decreasing the entropy simultaneously. The results also illustrate another phenomenon of tree coding: that tree coding becomes less effective in improving performance as the performance of the base algorithm improves.

More results were obtained for the first second of Sentence 1 using a 19 level quantizer. The results are shown in Table 5.2. These results again show that, in general, increasing L or M (or both) improves performance, but that the amount of improvement is smaller than that achieveable by the poorer-performing five level quantizer. There are, however, several instances where it appears that increasing L or M has decreased performance. This can occur because tree coding can make a suboptimal decision because the set of possible decisions is deliberately limited. As a result, tree coding can occasionally get "fooled".

Simulations were also performed with the tree coding ARC algorithm embedded within a pitch extraction loop. Some typical results are shown in Table 5.3. These results were obtained from Sentence 1, using an 11 level quantizer and M=5. Performance is again generally improved by tree coding, but by only a small amount.

Table 5.1. Results for a five level quantizer, M=5.

L	SNR (dB)	ENTROPY (BITS/SAMPLE)
1	14.03	1.423
2	14.60	1.421
4	14.66	1.407
8	14.98	1.403
16	15.08	1.393

Table 5.2. Results for a 19 level quantizer.

M \ L	1	2	3	4	5	6	7	8	9	10
1	21.30 2.263	21.30 2.262	21.30 2.262							
2	21.30 2.263	21.45 2.258	21.57 2.250	21.69 2.250	21.61 2.265	21.88 2.244	21.89 2.242	21.89 2.248	21.71 2.245	21.80 2.246
3	21.30 2.263	21.45 2.258	21.57 2.250	21.72 2.246	21.90 2.249					
4		21.45 2.258	21.63 2.248	21.63 2.243	21.89 2.244					
5		21.44 2.257	21.64 2.253	21.75 2.244	21.75 2.243					

Key: upper number is SNR (dB), lower number is entropy (bits/sample)

Table 5.3. Results for an 11 level quantizer, with M=5, embedded in a pitch removal loop.

L	SNR inside pitch loop (dB)	SNR overall (dB)	Entropy (bits/sample)
1	13.75	18.41	1.372
2	14.01	18.63	1.355
3	14.29	18.89	1.369
4	14.36	18.97	1.353
5	14.22	18.83	1.350
6	14.42	19.05	1.353

5.3 Adaptive Tree Coding

5.3.1 Description

Adaptive tree coding was developed as an attempt to gain the improvement in performance of the (M,L) algorithm, without using as many computations. It was felt that the performance improvement was desirable, but the large number of computations was a costly trade-off, and, more importantly, unable to be done in real time. This lead to the search for some way of improving the performance/computation ratio.

Adaptive tree coding was the result of that search. It is based on the fact that the receiver does not need to know to what extent, if any, that tree coding is being performed at the transmitter. The receiver acts only upon the quantizer levels it receives; it does not matter how those quantizer levels were arrived at. So the basic concept of adaptive tree coding is to use tree coding only when it appears to make sense.

Two strategies for adaptive tree coding were developed from this basic concept. The first strategy was to perform additional tree pruning, so that a node would have to have a reasonable chance of being selected to be kept. Specifically, a node would be pruned if the value of the criterion for it were worse than the value of the criterion for the best node, multiplied by an arbitrary factor. In this way, when growing the next level of nodes, time would not be spent growing nodes which had a high probability of being pruned eventually.

The second strategy was to "turn off" the coding (by setting M equal to 1) when the system was performing well, and "turn it back on" when it was not performing well. The idea behind this strategy was that if the system were

performing well, there was little to be gained from tree coding. This strategy was implemented by checking the value of the criterion for the best new node was better than some arbitrary threshold. If, on the other hand, tree coding was not in use, then tree coding would not be used until the value of the criterion for the best new node was worse than some second arbitrary threshold.

Both of these strategies were employed in our simulation of adaptive tree coding, because they are somewhat complementary in nature: the second strategy provides "course tuning", and the first strategy provides "fine tuning".

5.3.2 Results

Results were first obtained for adaptive tree coding using only the second ("threshold") strategy. Some typical results from these runs are shown in Table 5.4. Tree coding was used for 26% of the samples in Sentence 1, and for 14% of the samples in Sentence 11. The results indicate the ability of adaptive tree coding to improve performance with a small increase in computation.

Results were next obtained for adaptive tree coding using only the first ("factor") strategy. A sample of the results are shown in Table 5.5. On these runs, Sentence 1 was used as the input, and M was set to 5. A measure of the decrease in computations is "effective M", which is the average number of nodes retained for each new sample. It can clearly be seen that the number of calculations can be decreased dramatically while still retaining much of the increased performance. In fact, it can be seen from the data that the performance was improved by adaptive tree coding. One possible explanation for this phenomenon would be that for those cases where adaptive tree coding increased performance, that the additional pruning eliminated nodes which appeared to be good, but were not in the long run.

Finally, a series of simulation runs were made which utilized both adaptive strategies simultaneously. The results indicated that the two strategies were complementary, in that use of both was better than the use of either alone. Some representative results are shown in Table 5.6. It can be seen that about the same performance is obtained by adaptive tree coding as with tree coding, with about a 66% reduction in computations.

Table 5.4 **Results using the "threshold" strategy for adaptive tree coding.**

Sentence 1 (male speaker):

	PARC	Adaptive tree coding	Tree coding
SNR inside pitch loop (dB)	13.77	14.13	14.42
SNR overall (dB)	18.42	18.75	19.04
Entropy (bits/sample)	1.374	1.374	1.356

Sentence 11 (female speaker):

	PARC	Adaptive tree coding	Tree coding
SNR inside pitch loop (dB)	12.84	13.35	13.64
SNR overall (dB)	20.88	21.40	21.70
Entropy (bits/sample)	1.386	1.374	1.338

Table 5.5 Results using the "factor" strategy for adaptive tree coding.

FACTOR	Overall SNR(dB)	Entropy (bits/sample)	Effective M
1.E6 (60dB)	18.92	1.347	5.00
10.00(10dB)	18.92	1.347	4.46
7.94(9dB)	18.92	1.348	4.43
6.31(8dB)	18.92	1.348	4.39
5.01(7dB)	18.92	1.348	4.33
3.98(6dB)	18.92	1.348	4.24
3.16(5dB)	18.94	1.349	4.10
2.51(4dB)	19.09	1.343	3.84
2.00(3dB)	19.08	1.348	3.41
1.58(2dB)	19.07	1.356	2.62
1.50(1.75dB)	19.05	1.354	2.40
1.41(1.50dB)	18.94	1.360	2.10
1.33(1.25dB)	18.94	1.347	1.85
1.26(1dB)	18.82	1.354	1.58
1.00(0dB)	18.42	1.374	1.00

Table 5.6 Results using both strategies for adaptive tree searching.

	Overall SNR(dB)	Entropy (bits/sample)	Effective M
PARC	18.42	1.374	1.00
Adaptive tree coding HITHR = 2416 (15dB), LOTHR = 304.1 (24dB), FACTOR = 1.50 (1.75dB)	18.92	1.360	1.51
Tree coding	18.92	1.347	5.00

5.4 Conclusions and suggestions for further research

It would appear that on the basis of the results obtained, tree coding is an effective way to increase the performance of a quantization system. The major question though, is not whether it is effective, but how effective it is relative to what it costs. It would appear that tree coding may not be effective in this sense, but that some form of adaptive tree coding may be. The question was moot for this project, because there was not sufficient real time to perform a tree coding version of PARC.

The reason why tree coding appears to be relatively ineffective with PARC may be that the ARC algorithm has been overly adapted for PARC — that is a set of parameters optimized for PARC will most probably be a poor choice to be used with tree coding. This makes a lot of sense if you think about how tree coding can help a system. For the PARC algorithm to work well, the parameters are chosen so that typically only one quantizer level results in a reasonable amount of error — to do otherwise would be suboptimal for PARC, since only one quantizer level can be selected, and having more than one reasonable quantizer level would simply reduce the dynamic range of the quantizer or increase the average quantization error. In contrast, for tree coding to be effective, several quantizer levels should be reasonable for each sample.

In order to modify the ARC parameters to make tree coding more effective then, it would appear that it would be desirable to select parameters which would decrease granular noise and let the tree coding reduce slope overload noise. Specifically, it would seem to be desirable to "move in" the output and scaling factors of the quantizer, decrease the updating gains, and reduce the "time constants" of the system. In this way, there would be a

richer selection of quantizer levels for the tree code.

There are also several other areas to be researched. In our simulations, the tree coding took place entirely within the pitch loop. It might be interesting, therefore, to investigate the effect of basing the pruning criterion on the reconstructed speech rather than the reconstructed depitched speech. Doing this might aid in the smooth transition from one pitch block to the next. Another idea which could be investigated would be delayed updating of the ARC algorithm. In delayed updating, the algorithm would be updated on the basis of the quantizer level just decided upon (corresponding to the sample L samples earlier), so that the updating could be done once for all nodes, eliminating many computations.

Tree coding might also be made more effective by making changes in the tree coding algorithm itself. One possible path for investigation would be in the area of variable symbol release, as developed by Goris. Another possible path would involve investigating different forms of the pruning criterion. For example, it might improve performance to weigh the contribution from earlier samples more heavily than that from recent samples, because it would appear that the "soft" decisions become "harder" as time passes.

5.5 References

1. J. B. Anderson and J. B. Bodie, "Tree Encoding of Speech", IEEE Trans. on Information Theory, Vol. IT-21, No. 4, July 1975, pp. 379-387.

2. J. B. Anderson and C.-W. P. Ho, "Architecture and Construction of a Hardware Sequential Encoder for Speech", IEEE Trans. on Communications, Vol. COM-25, No. 7, July 1977, pp. 703-707.

3. J. D. Gibson and A. C. Goris, "Incremental and Variable-Length Tree Coding of Speech", Proceedings 1979 International Conference on Communications, Boston, MA, June 1979.

4. A. D. Goris, "Tree Coding of Speech Using Variable Length Symbol Release Techniques", Master's Thesis, Texas A&M University, August 1979.

5. N. S. Jayant and S. A. Christensen, "Tree-Encoding of Speech Using the (M,L)-Algorithm and Adaptive Quantization", IEEE Trans. on Communications, Vol. COM-26, No. 9, September 1978, pp. 1376-1379.

6. J. Uddenfeldt and L. H. Zetterberg, "Algorithms for Delayed Encoding in Delta Modulation with Speech-Like Signals", IEEE Trans. on Communications, Vol. COM-24, No. 6, June 1976, pp. 650-655.

7. S. G. Wilson and S. Husain, "Adaptive Tree Encoding of Speech at 8000 Bits/s with a Frequency-Weighted Error Criterion", IEEE Trans. on Communications, Vol. COM-27, No. 1, January 1979, pp. 165-170.

8. L. H. Zetterberg and J. Uddenfeldt, "Adaptive Delta Modulation with Delayed Decision", IEEE Trans. on Communications, Vol. COM-22, No. 9, September 1974, pp. 1195-1198.

CHAPTER 6

BACKWARD PITCH EXTRACTION ADAPTIVE RESIDUAL CODER (BPARC)

6.1 Introduction

Many of the practical systems for digitizing speech are variants of differential pulse code modulation (DPCM). The speech coder developed for the 9.6 Kbs bit rate uses this structure augmented by a pitch extraction loop. This algorithm is called Pitch extraction Adaptive Residual Coder (PARC).

The system described in the chapter is identical to PARC except for the method of pitch extraction. In PARC, pitch is extracted block by block from raw speech. Once the correlation coefficient β and pitch period T are known, reduced speech is formed, processed by the coder and transmitted. The β's and T's also need to be transmitted; the number of bits required to transmit these parameters depends on the number of parameters available (this depends on block length), type of coding employed and the way they are transmitted. In addition to the number of bits required for transmission, the transmission of these parameters necessitates a framing of the bit stream and an associated frame synchronization problem.

In order to avoid this framing problem, a backward adaptive approach was investigated which would not require transmission of β's and T's. Since the values of β and T change very little in a given voiced region, these parameters can be calculated for a previous block of speech and used for the current block to reconstruct speech. As noted in an earlier chapter, short pitch blocks provide the best performance. However, these short blocks require the transmission of a large amount of side information,

β and T's, making them impractical. The use of the backward identification of β and T eliminates this problem.

In following sections, this approach referred to as Backward PARC (BPARC) is described along with various computer simulation studies comparing it with the standard PARC algorithm. A complete listing of the source program and a flow chart for the simulation are also included in this chapter.

6.2 System Structure

Figure 6.1 shows the block diagram of the backward adaptive pitch extraction residual coder. A comparison of this figure with Fig. 2.1 reveals the obvious difference that β and T are now computed from $\hat{s}(k)$'s rather than s(k)'s. As a result of this change, it is no longer necessary to transmit β and T since, in case of no transmission errors, the receiver can carry out the same computation. Note that the receiver must now be more complex since it must be capable of computing β and T.

The use of $\hat{s}(k)$ to compute β and T causes another less obvious change in the transmitter. In order to use $\hat{s}(k)$'s, the computation of β and T must be based only on past speech estimates. In the forward adaptive case, the computation of β and T is based on both future and past speech samples. The basic approach for BPARC is to compute β and T on a block of $\hat{s}(k)$, $k = k_o, k_o-1,\ldots,k_o - K_c$ and then to use this value of β and T to form reduced speech v(k) for $k = k_o+1, k_o+2,\ldots, k_o+K_u$. Here K_c is called the computation block size and K_u is the use block size; these two values are not necessarily equal.

The basic philosophy of the BPARC approach is that in voiced segments, where pitch redundancy removal is most effective, β and T does not change rapidly. Table 6.1 illustrates this fact with a listing of β and T for a typical segment of voiced speech. The use of a β and T, computed for one block, in the next block will not have much effect on the performance in such a voiced segment.

There will, however, be rapid changes in β and T during transitions from voiced to unvoiced or unvoiced to voiced speech. During these transitions, significant performance degradation can be expected in BPARC

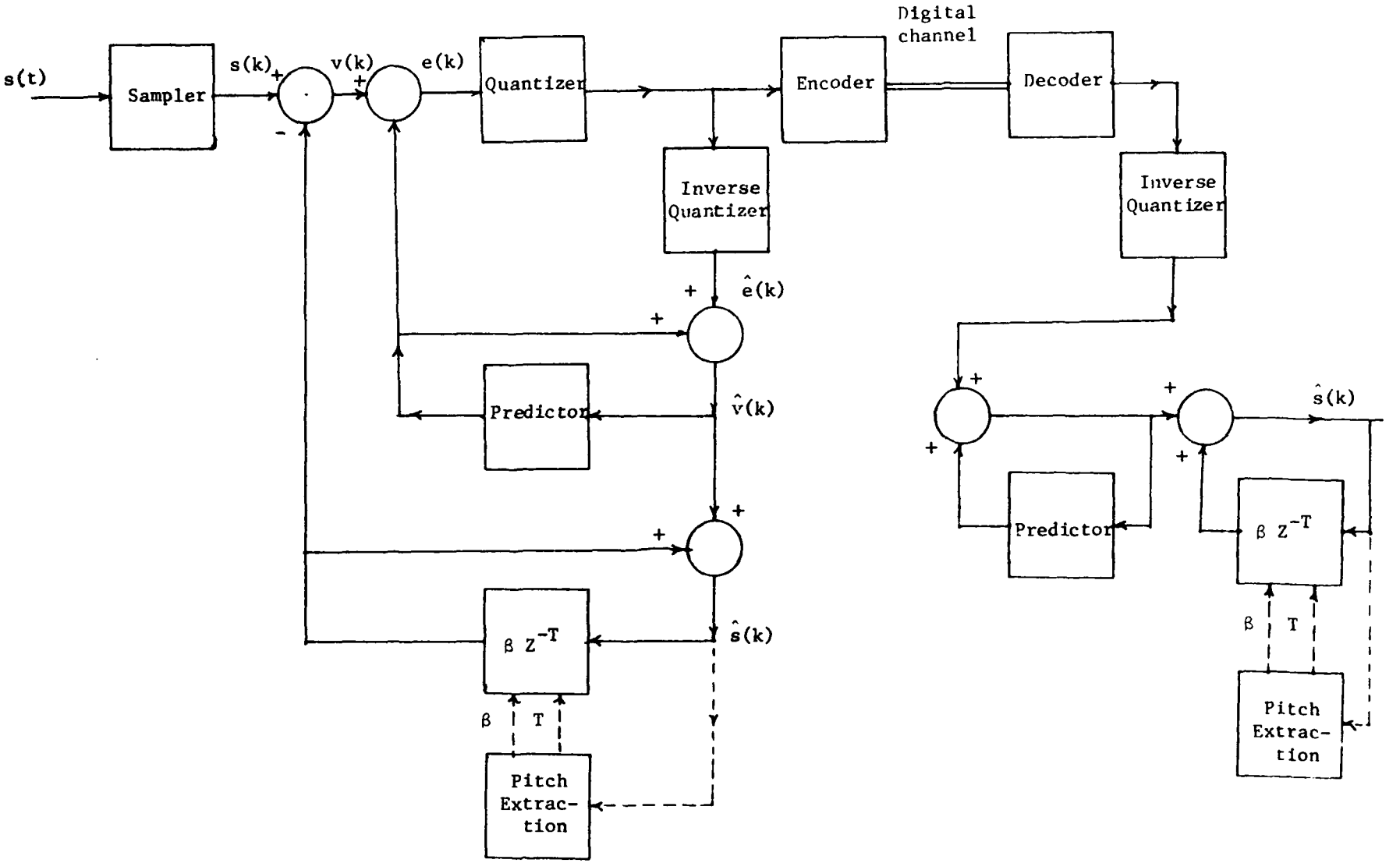


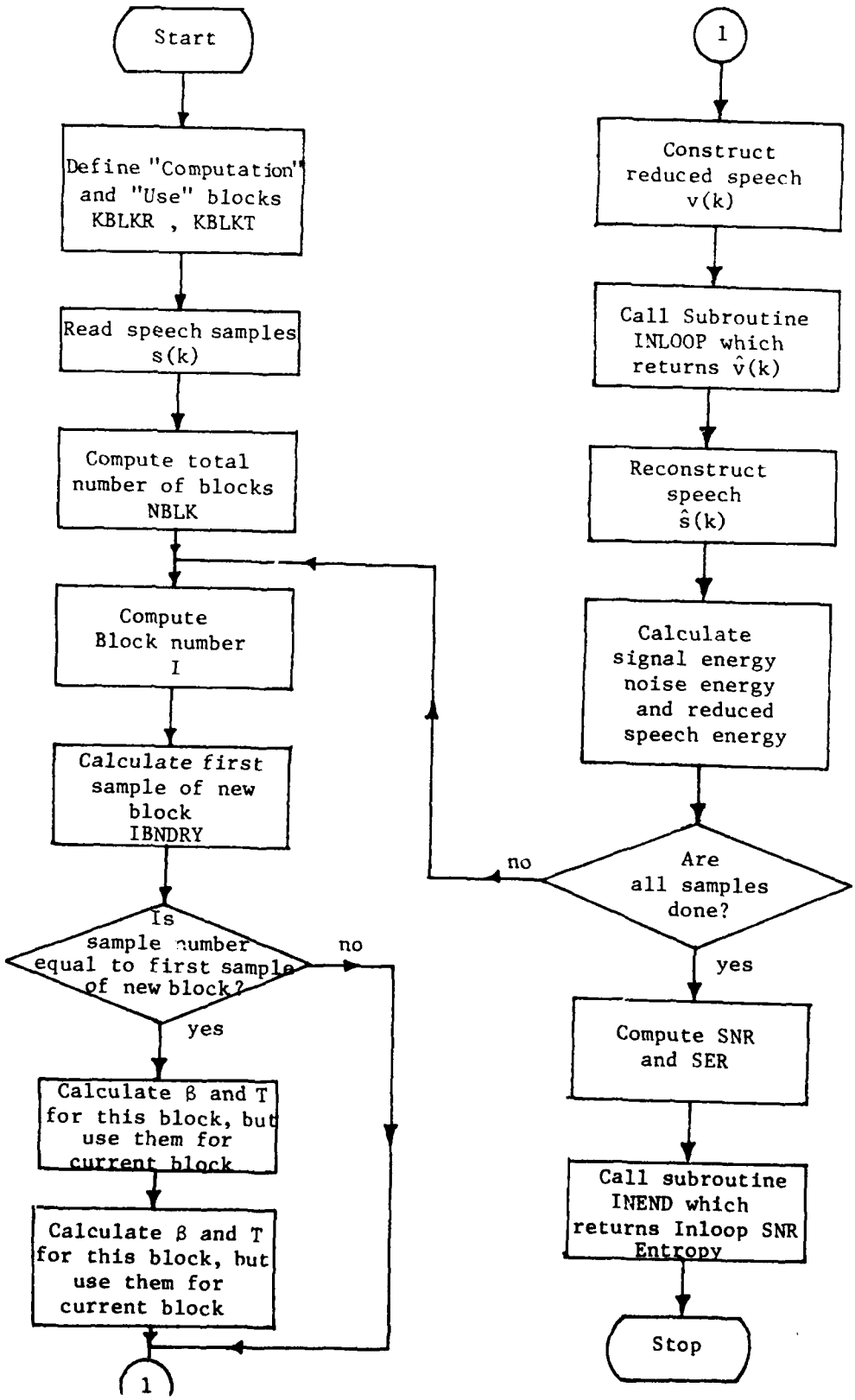
Fig. 6.1 Backward Pitch Extraction Adaptive Residual Coder

Table 6.1

Values of β and T for a Segment of Voiced Speech

Sample Number	β	T
5400	0.91	30
5520	0.80	30
5550	0.86	30
5500	0.92	30
5610	0.99	30
5640	0.95	30
5670	1.04	30
5700	0.98	30
5730	0.98	30
5760	1.02	30
5790	1.03	30
5820	1.00	30
5850	1.02	30
5880	1.00	30
5910	1.06	30
5940	1.06	30
5970	1.04	30
6000	1.09	30
6030	1.08	30
6060	1.09	30
6090	1.05	30
6120	1.06	30
6150	1.06	30
6180	1.10	30
6210	1.06	30
6240	1.06	30
6270	1.03	30
6300	1.02	30
6330	1.02	30
6360	1.00	30
6390	1.00	30
6420	0.98	30
6450	0.99	30
6480	0.97	30
6510	0.95	30
6540	0.93	30
6570	0.94	30
6600	0.94	30
6630	0.87	30
6660	0.89	30
6690	0.85	30
6720	0.84	30

Fig. 6.2 Backward Pitch extraction Adaptive Residual Coder (BPARC)



over the PARC algorithm. The hope was that the decrease in quantization noise caused by removing the β and T transmission would offset this degradation. The advantage of not requiring framing for β and T is also obvious.

Exactly the same method, namely, Eqs. (2.6) and (2.7) are used to compute β and T for the BPARC as for the standard PARC except $\hat{s}$ is used in place of s and all $\hat{s}$'s are past values. Hence T at stage k is the value of τ which minimizes the AMDF function given by

$$A(\tau) = \frac{1}{K_c} \sum_{j=k-K_c}^{k} |\hat{s}(j)-\hat{s}(j-\tau)| \qquad (6.1)$$

$$\tau = 20,21,\ldots\tau_{max}$$

Once T is known, β is determined from

$$\beta = \frac{\sum_{j=k-K_c}^{k} \hat{s}(j)\hat{s}(j-T)}{\sum_{j=k-K_c}^{k} \hat{s}(j-T)\hat{s}(j-T)} \qquad (6.2)$$

A block diagram for the complete BPARC algorithm is given in Fig. 6.2. This algorithm assumes that a complete sentence is read in and then processed. This algorithm was programmed on the PDP-11/60 in order to determine its performance characteristics.

6.3 Performance Evaluation and Parametric Studies

As the first step in evaluating the performance of the BPARC algorithm, a comparison of β, T and SER using the normal (forward) PARC and the BPARC algorithms was made. Table 6.2 shows a typical portion of the results from this study. Note that SER is significantly degraded in the transition regions at the beginning and end. There is some decrease (~1 db) in SER in the voiced segment. A computation block length of 30 samples was used for both cases. This degradation in SER is understandable since we use β and T which were calculated from previously processed speech are used for the present block of samples. In transition regions such as V/UV, UV/S, S/V, the previously processed speech block can be much different than the current block of speech to be processed.

A study of the effect of different lengths of "computation block" and "use block" on SNR was conducted next. In general for very large "computation" or "use" block SNR goes down while for smaller block lengths increase in performance was noticed. However, there is not a monotonic increase or decrease in performance noticed with a decrease or increase in block lengths. The results are tabulated in Table 6.3.

From the very basic idea of BPARC, it is clear that this approach should work better for voiced speech. Exactly the same thing was observed when the range of possible β's was limited to correspond to voiced speech. Table 6.4 shows the improvement in SNR by restricting β.

Table 6.4

Effect of Limit Range for β
(Sent 11, Female Speaker, $K_u = K_c = 30$)

	$-2 < \beta < 2$	$0.74 < \beta < 1.14$
SNR	19.96 db	20.69 db
SNR(inloop)	15.00 db	14.28 db
SER	4.96 db	6.41 db
H	1.51 b/sample	1.46 b/sample

Table 6.2

Comparison of β,T and SER for PARC and BPARC (Sent 11)

Sample number	PARC β	PARC T	PARC Signal Energy Reduction	BPARC β	BPARC T	BPARC Signal Energy Reduction
5310	0.37	29	1.11 DB	0.00	23	0.00
5340	2.00	26	4.36 DB	0.00	23	0.00
5370	1.32	26	7.71 DB	0.00	26	0.00
5400	0.81	28	8.90 DB	0.00	26	0.00
5430	0.93	29	13.54 DB	0.89	27	2.54
5460	0.90	29	11.90 DB	0.85	28	3.67
5490	0.91	30	16.55 DB	0.91	29	5.05
5520	0.80	30	15.54 DB	0.88	30	14.72
5550	0.86	30	10.35 DB	0.87	30	10.94
5580	0.92	30	14.90 DB	0.88	30	14.94
5610	0.99	30	15.70 DB	0.90	30	15.57
5640	0.95	30	13.46 DB	0.95	30	14.82
5670	1.04	30	18.42 DB	0.95	30	16.37
5700	0.98	30	16.69 DB	0.99	30	16.62
5730	0.98	30	14.80 DB	1.01	30	15.16
5760	1.02	30	17.09 DB	0.96	30	19.33
5790	1.03	30	17.40 DB	0.99	30	20.15
5820	1.00	30	17.30 DB	1.03	30	16.39
5850	1.02	30	19.08 DB	1.01	30	18.69
5880	1.00	30	22.39 DB	1.02	30	19.72
5910	1.06	30	20.45 DB	0.99	30	18.55
5940	1.06	30	15.61 DB	1.02	30	16.58
5970	1.04	30	18.70 DB	1.07	30	18.96
6000	1.09	30	17.62 DB	1.07	30	17.03
6030	1.08	30	16.16 DB	1.06	30	16.07
6060	1.09	30	16.66 DB	1.07	30	19.13
6090	1.05	30	15.84 DB	1.09	30	15.74
6120	1.06	30	13.82 DB	1.06	30	14.31
6150	1.06	30	11.71 DB	1.08	30	12.02
6180	1.10	30	15.62 DB	1.08	30	15.72
6210	1.06	30	16.46 DB	1.08	30	16.42
6240	1.06	30	14.30 DB	1.08	30	13.59
6270	1.03	30	14.53 DB	1.05	30	14.03
6300	1.02	30	16.95 DB	1.05	30	17.52
6330	1.02	30	17.48 DB	1.03	30	16.48
6360	1.00	30	17.05 DB	1.01	30	17.34
6390	1.00	30	21.79 DB	1.01	30	21.29
6420	0.98	30	17.56 DB	1.00	30	18.34
6450	0.99	30	20.76 DB	0.99	30	19.43
6480	0.97	30	18.49 DB	0.98	30	18.26
6510	0.95	30	18.88 DB	0.97	30	19.37
6540	0.93	30	13.91 DB	0.96	30	12.93
6570	0.94	30	12.86 DB	0.94	30	13.94
6600	0.94	30	12.88 DB	0.92	30	12.68
6630	0.87	30	13.60 DB	0.95	30	12.51
6660	0.89	30	12.98 DB	0.89	30	14.22
6690	0.85	30	10.81 DB	0.88	30	10.95
6720	0.84	30	14.74 DB	0.87	30	14.87
J	0.		9.9 B	0.8		9
6780	0.76	30	8.72 DB	0.83	30	8.8

Table 6.3

Variation of Use and Computation Block Lengths

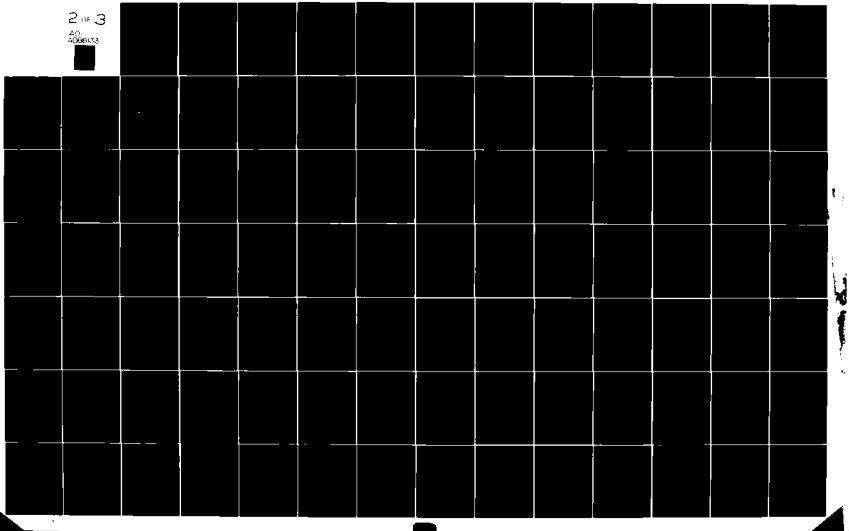
COMPUTATION BLOCK LENGTH K_c (Samples)	USE BLOCK LENGTH K_u (Samples) 20	30	40	50
20	19.06* 15.05 4.01 1.53	20.05 14.10 5.96 1.51		
30	19.23 14.62 4.61 1.51	19.96 15.00 4.96 1.51	18.80 15.59 3.22 1.50	
40	19.25 15.11 4.15 1.48	19.43 14.82 4.61 1.49	19.15 15.27 3.88 1.50	19.39 14.76 4.62 1.49
50	19.46 14.91 4.55 1.49	20.06 14.12 5.94 1.50	19.22 15.02 4.21 1.50	18.34 14.97 3.37 1.50
60	19.19 14.98 4.21 1.49	19.64 14.56 5.08 1.49	19.17 15.05 4.11 1.51	18.89 15.13 3.75 1.50
70	19.24 14.90 4.35 1.51	19.53 14.93 4.60 1.48	19.05 15.61 3.44 1.48	18.43 15.00 3.43 1.51
80		19.80 14.84 4.96 1.49	18.61 15.06 3.55 1.52	18.12 14.86 3.26 1.51
90			18.90 15.09 3.81 1.49	18.57 15.29 3.28 1.51

*Each block lists SNR (overall), SNR (inloop), SER and H.

AD-A086 133 NOTRE DAME UNIV IN DEPT OF ELECTRICAL ENGINEERING F/G 17/2
DESIGN AND IMPLEMENTATION OF A SPEECH CODING ALGORITHM AT 9600 --ETC(U)
APR 80 J L MELSA, D L COHN, A ARORA DCA100-79-C-0005

UNCLASSIFIED NL

2 OF 3
AD A086133



In the above approach, β was set zero if it was not in the allowed range. It appeared that the resulting sharp changes in β would degrade performance. To eliminate this problem a limit was set on the maximum percentage changes in β so that β would vary more smoothly in transition regions. This approach helped to increase SER as shown in Table 6.5 but not the performance as SNR inloop decreased.

Table 6.5

Effect of Limiting $\Delta\beta$
(Sent 1, male speaker, $K_c = 45$, $K_u = 30$), $0.65 \leq \beta \leq 1.25$)

	$\Delta\beta = 0$	$\lvert\Delta\beta\rvert < 10\%$	$\lvert\Delta\beta\rvert < 20\%$
SNR	17.78 db	17.57 db	17.70 db
SNR(inloop)	14.84 db	14.45 db	14.67 db
SER	2.94 db	3.12 db	3.02 db
H	1.49 b/sample	1.50 b/sample	1.49 b/sample

To calculate the pitch parameter T the function

$$A(T) = \sum_{j=1}^{K} |\hat{s}(j) - \hat{s}(j-T)| \quad \text{was}$$

minimized with respect to T for $20 \leq T \leq L$ where $L \geq K_c$ computation block length = 30. Different values of L were tried. Some improvement in performance was noticed (see Table 6.6) if search range was decreased up to a certain point. For a long search range, the effect of errors that were made while reconstructing previous samples becomes more severe while a small search range may not be enough to detect correct period T.

Table 6.6

Effect of Search Range for T
(Sent 1 : male speaker, $K_u = K_c = 30$)

	$20 \leq T \leq 100$	$20 \leq T \leq 80$	$20 \leq T \leq 70$
SNR	17.67 db	17.78 db	17.78 db
SNR(inloop)	14.87 db	14.84 db	14.75 db
SER	2.80 db	2.94 db	3.03 db
H	1.49 b/sample	1.49 b/sample	1.49 b/sample

As discussed eqrlier for the normal PARC algorithm it is necessary to transmit β's and T's along with quantized residual reduced speech. This, of course, requires a few bits per sample. If a block size of 100 is used and β and T require 13 bits, this becomes 0.13 bits/sample. That leaves 1.37 bits/sample for transmission of other information. Hence, to make a fair comparison between the two algorithms, H was limited to 1.37 for PARC and 1.5 bits/sample for BPARC. Even though BPARC is not a clear winner (see Table 6.7), it is a very attractive solution to problems of transmitting β and T.

Table 6.7
Comparison of PARC and BPARC Algorithms

		SNR	SNR inloop	SER	H
Sentence 1	PARC	19.11 db	13.63 db	5.48 db	1.37 b/sample
	BPARC	17.78 db	14.84 db	2.94 db	1.49 b/sample
Sentence 11	PARC	21.40 db	12.95 db	8.45 db	1.36 b/sample
	BPARC	20.53 db	14.21 db	6.31 db	1.47 b/sample

6.4 Transmission Error Studies

No speech encoding algorithm is good unless it can tolerate with channel errors. To study the effects of channel errors, first step is to determine the effect of errors in quantizer levels. The errors that are introduced in quantizer levels do not exactly correspond to errors caused due to bit reversal but nevertheless they are a good measure of the algorithm's susceptibility to channel errors.

The following procedure was adopted to study effect of channel errors in the BPARC algorithm:

(i) The receiver program was extracted from transmitter program.

(ii) A quantizer output file was created with desired transmission errors.

(iii) This file was read by receiver and a reconstructed speech $\hat{s}(k)$ file was created.

(iv) SNR was calculated between original speech and received speech.

The algorithm tolerated one transmission error (for male speaker SNR goes down from 19.52 to 19.48) but becomes unstable with additional errors. It was suspected that wrong $\hat{s}$'s lead to wrong β's and possibly wrong T's which in turn make $\hat{s}$'s wrong. To investigate the cause, the BPARC program was run with no pitch removal. The algorithm works very well (see Table 6.8) even with 1% transmission error rate thus confirming above doubt.

Table 6.8

Effect of Transmission Errors on BPARC with no Pitch Extractor
(Sentence 1: Male speaker,
SNR with no transmission error = 18.83 db, (H=2.18)

Transmission error rate	SNR
0.01%	18.82 db
0.1%	18.37 db
1%	4.25 db

A fixed predictor instead of adaptive predictor was also tried in the system. It was found that fixed predictor minimizes the effect of transmission errors to some extent.

Table 6.9 lists β's and T's with and without transmission error. Note that β changes immediately after error has been introduced. The plots in Figs. 6.3 and 6.4 compare the effect of transmission error on local SNR.

Table 6.9

Effect of Transmission Error (Sentence 11: Female speaker)

Sample Number	Without transmission error β	T	With transmission error at sample number 1674, 7075 β	T
1290	0.88	27	0.88	27
1320	0.94	27	0.94	27
1350	0.97	55	0.97	55
1380	0.98	27	0.98	27
1410	0.98	27	0.98	27
1440	0.95	27	0.95	27
1470	0.99	27	0.99	27
1500	0.99	27	0.99	27
1530	0.92	27	0.92	27
1560	0.94	27	0.94	27
1590	0.94	27	0.94	27
1620	0.97	55	0.97	55
1650	0.98	55	0.98	55
1680	1.01	55	1.01	55
1710	1.00	55	0.99 ←	55
1740	1.02	55	1.02	55
1770	1.02	55	1.02	55
1800	1.00	55	0.99	55
1830	0.93	55	0.93	55
1860	0.89	55	0.88	55
1890	0.81	27	0.80	27
1920	0.83	27	0.82	27
1950	0.87	27	0.86	27
1980	0.91	27	0.91	27
2010	0.91	27	0.91	27
2040	0.90	27	0.91	27
2070	0.90	28	0.91	28
2100	0.87	28	0.89	28
2130	0.00	28	0.65	28
2160	0.70	57	1.02	57
2190	0.70	28	1.01	57
2220	0.00	27	0.85	57
2250	0.00	24	0.84	57
2280	0.00	25	0.84	57
2310	0.00	28	0.84	57
2340	0.00	28	0.84	57
2370	0.00	28	0.84	57
2400	0.00	28	0.84	57
2430	0.00	28	0.84	57
2460	0.00	28	0.84	57
2490	0.00	28	0.84	57
2520	0.00	28	0.84	57
2550	0.00	28	0.84	57
2580	0.00	28	0.84	57
2610	0.00	28	0.84	57
2640	0.00	28	0.84	57
2670	0.00	28	0.84	57
2700	0.00	28	0.84	57
2730	0.00	28	0.84	57

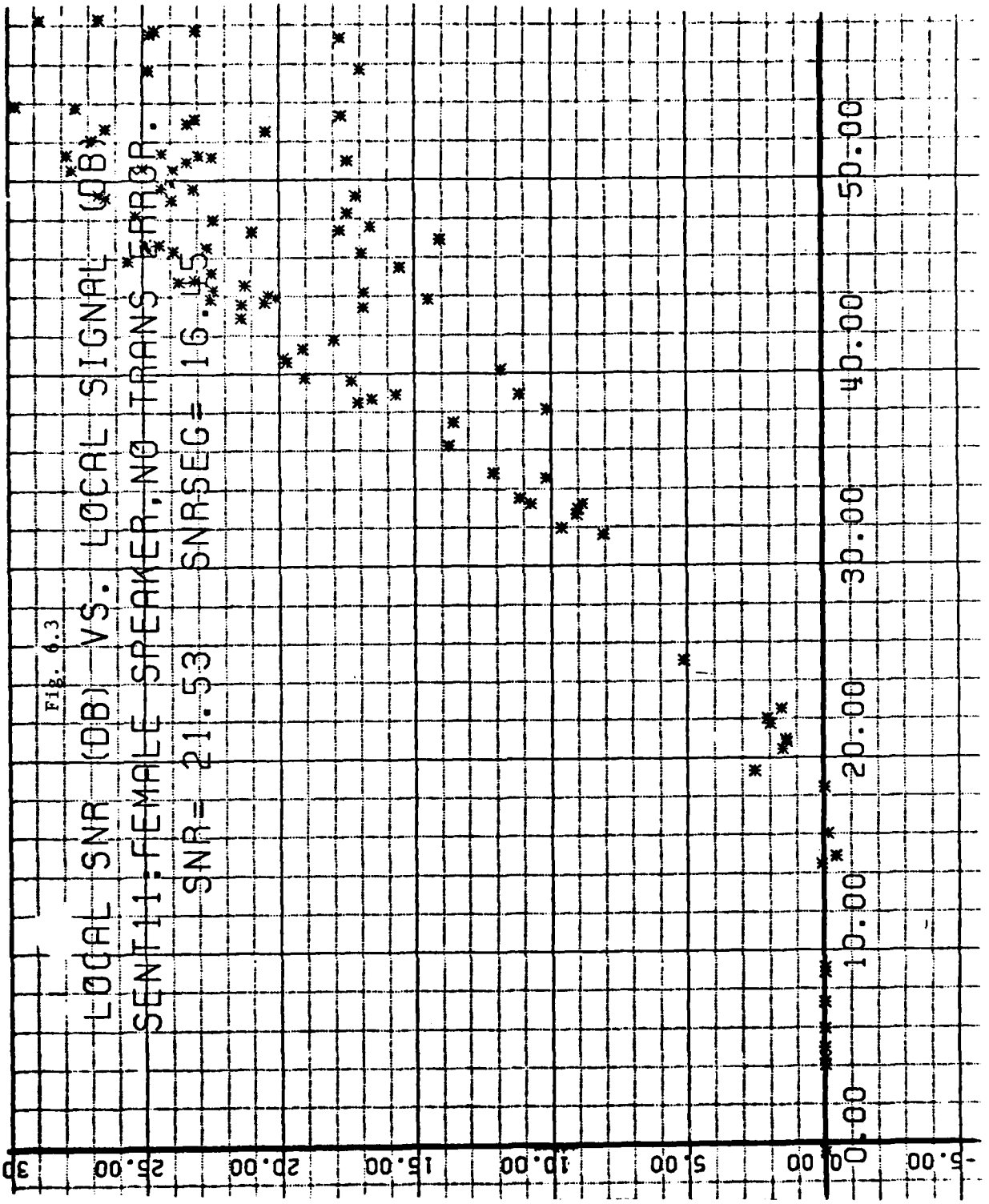

LOCAL SNR (DB) -VS- LOCAL SIGNAL (DB)
SENT11:FEMALE SPEAKER,NO TRANS ERROR.
SNR= 21.53 SNRSEG= 16.45
-5.00
0.00
5.00
10.00
15.00
20.00
25.00
30
-10.00
0.00
10.00
20.00
30.00
40.00
50.00

Fig. 6.3

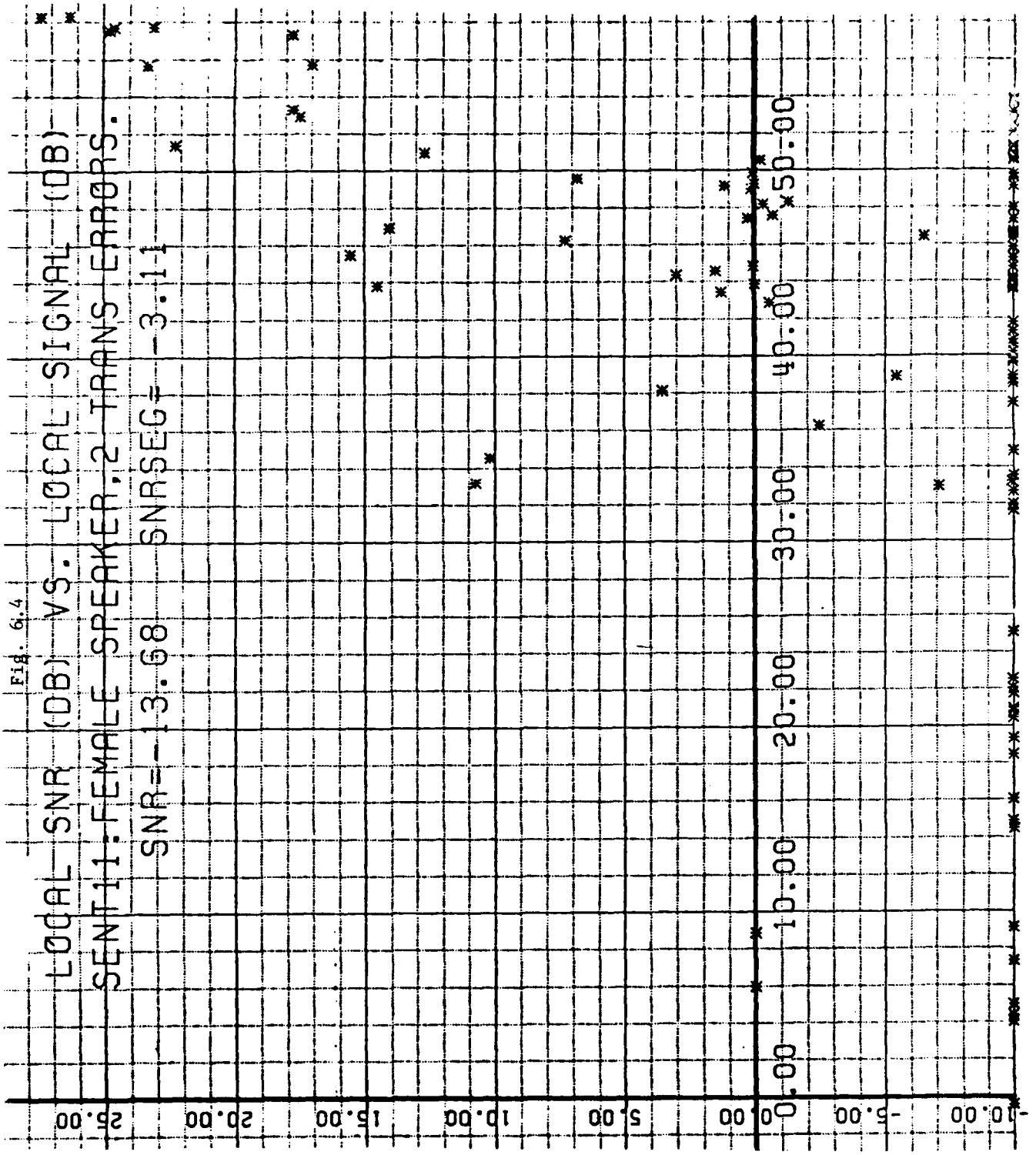

LOCAL SNR (DB) VS. LOCAL SIGNAL (DB)
SENT11:FEMALE SPEAKER,2 TRANS ERRORS.
SNR=-13.68 SNRSEG=-3.11
-10.00
-5.00
0.00
5.00
10.00
15.00
20.00
25.00
10.00
20.00
30.00
40.00
50.00
dB

Fig. 6.4

6.5 Conclusions

The study presented in this chapter of the BPARC algorithm shows that it is an attractive approach for a fairly error-free channel but not suitable for channel with at least 0.1% error rate. The following changes in algorithm might help to make it more robust.

1. Currents β and T are calculated for computation block (using received speech) and then used for the current block to construct speech. It is now possible to use the new received speech samples for current block and re-calculate β and T using those samples. These values of β and T should be closer to true values for that block.
2. Maximum changes in β could be limited thus reducing effect of channel error.
3. Errors in pitch periods (in voiced region) due to channel errors could be known by looking at T's of previous blocks and then could be corrected.

Unfortunately time did not permit an examination of these ideas.

CHAPTER 7
TANDEM OPERATION

7.1 Introduction

One of the requirements of the speech coding algorithm is that it should perform satisfactorily in tandem with a CVSD speech coder operating at a data rate of 16 Kb/s and this tandem configuration should provide speech intelligibility with minimal degradation compared with a single link of CVSD operating at 16 Kb/s. The simulation of this tandem operation was done and results are discussed in Sec. 7.2.

Another requirement of speech coding algorithm is that it should produce intelligible speech under acoustic background noise. The simulation of background noise and the performance of PARC are discussed in Sec. 7.3.

7.2 PARC in Tandem with CVSD

The performance of the PARC system in tandem with CVSD system was studied. This study included passing raw speech through CVSD algorithm to create a file of CVSD output speech and using this speech file as input to the PARC algorithm. The output of the PARC algorithm becomes the output of the tandem system CVSD-PARC as shown in Fig. 7.1a. Similarly the PARC-CVSD tandem connection shown in Fig. 7.1b was also studied. The CVSD algorithm used for the study of tandem operation is described in Appendix D.

The CVSD algorithm used in this study operates on input speech sampled at the rate of 16K samples per second. This means the sampling rates at the input and the output of CVSD must be modified in order to make the tandem connections with PARC which operates on speech sampled at 6.4 KHz. The resampling can be done by using resampling programs listed in Appendix D. The CVSD program has incorporated this resampling program which makes simulation less time consuming.

The performance of these tandem connections are judged by Signal to quantization Noise Ratio and the subjective criterion. Sentence 1, "Cats and Dogs each hate the other", spoken by a male speaker, was used for simulation. Results are reported in Table 7.1 and 7.2.

TABLE 7.1 SNR for PARC,CVSD and their interconnections.

PARC	CVSD	PARC-CVSD tandem	CVSD-PARC tandem
18.31 db	11.78 db	10.80 db	10.68 db

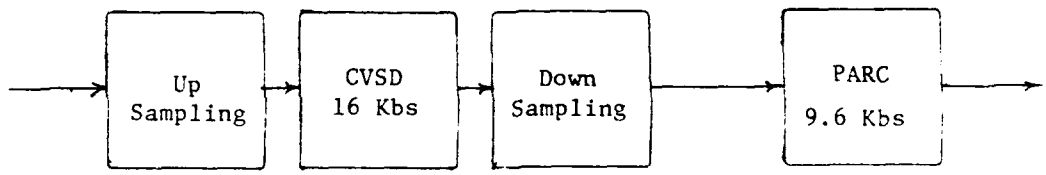


Fig. 7.1a CVSD in Tandem with PARC

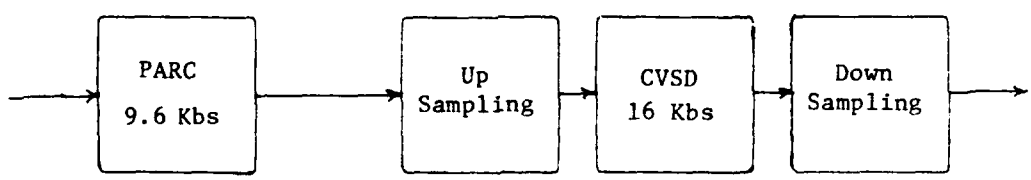


Fig. 7.1b PARC in Tandem with CVSD

TABLE 7.2 Effect of CVSD Speech input on PARC performance

	SNR	Inloop SNR	SER	Entropy H
PARC with raw speech as input	18.31 db	13.69 db	4.62 db	1.41 bits/sample
PARC with CVSD speech as input	16.84 db	12.90 db	3.94 db	1.74 bits/sample

The results in Table 7.1 show that there is little degradation in SNR due to tandem operations. In fact, the speech quality of CVSD-PARC tandem seems to be better than CVSD alone in terms of perception. This study indicated that PARC acts as a filter for the granular noise in the CVSD output. The performance of these tandem operations could be improved by redesigning various parameters used in PARC. However, at this point the purpose of the study was to make sure that PARC algorithm performs reasonably well in tandem with CVSD.

It can be seen from Table 7.2 that the SNR decreased by less than 2 db by inputting the CVSD speech instead of raw speech. This decrease in SNR could be attributed to the decrease in the predictor performance as a result of high frequency contents of CVSD speech and poor signal energy reduction due to decrease in correlation between CVSD speech samples. However, this decrease in SNR is surprisingly low. This shows that the PARC algorithm works very well for noisy input speech except for some increase in entropy. The entropy increase is a rather serious problem but solvable by using buffer control techniques which are discussed in details in Chapter 10.

7.3 Effect of Background Speakers on PARC Performance

It has been observed that PARC algorithm performs satisfactorily well in tandem with CVSD. However, the noise in CVSD speech is white. It is also important to study the effects of correlated background noise such as office noise.

One of the speech coding algorithm requirements is that the speech coder shall produce intelligible speech under conditions of acoustic background noise, 60 db referenced to 20 μ Newtons/meter^2. This statement, though technically precise, gives little feeling about the loudness of noise. Figure 7.2 [1] gives comparative intensities of variety of common sounds. The noise level described above is similar to quiet office noise. Regarding sound energy, Alex ander Woods'[2] quotation gives the whole picture. He, in his book, "Physics of Music", points out that sound energy generated by shouting of the crowd throughout an exciting game (say, 50,000 people at a 90 minute football match between Notre Dame and USC is just about enough to warm one cup of coffee.

The effect of background noise, consisting of typewriter noise, conversation, music and so forth was studied on the real-time system. The algorithm performs with no difficulty.

The study of background noise is very simple after the algorithm has been implemented in real time. It is just a matter of talking into handset with noise in the background. The output could be heard through headphone. However, in the FORTRAN simulation, the task is not so straightforward. There is a need for digital speech file with background noise. It was thought that periodic background noise would be the worst kind of noise for PARC algorithm. Therefore, it was decided to study the performance of the algorithm for multispeaker files. Multispeaker files were created by adding two digital speech files with appropriate weight.

In the simulation, the multispeaker file was generated by adding sentence 11 (female speaker) to male speaker, sentence 7, as shown in Eq. (7.1).

$$s_{composite} = S_{11} + k\, S_1 \tag{7.1}$$

where k takes values from 0 to 1 thus having varying degree of background noise. It was noticed that pitch extraction loop picks pitch for both the speakers and algorithm performs very well as can be seen from the results in Table 7.3.

TABLE 7.3 SNR's for Multispeaker Files

Multispeaker file	SNR	Inloop SNR	SER	Entropy H Bits/sample
$S_{11} + 0\, S_1$	21.11 db	13.01 db	8.10 db	1.39
$S_{11} + .25\, S_1$	20.16 db	13.03 db	7.13 db	1.57
$S_{11} + .5\, S_1$	19.44 db	13.39 db	6.05 db	1.66
$S_{11} + 1\, S_1$	18.92 db	13.95 db	4.97 db	1.76

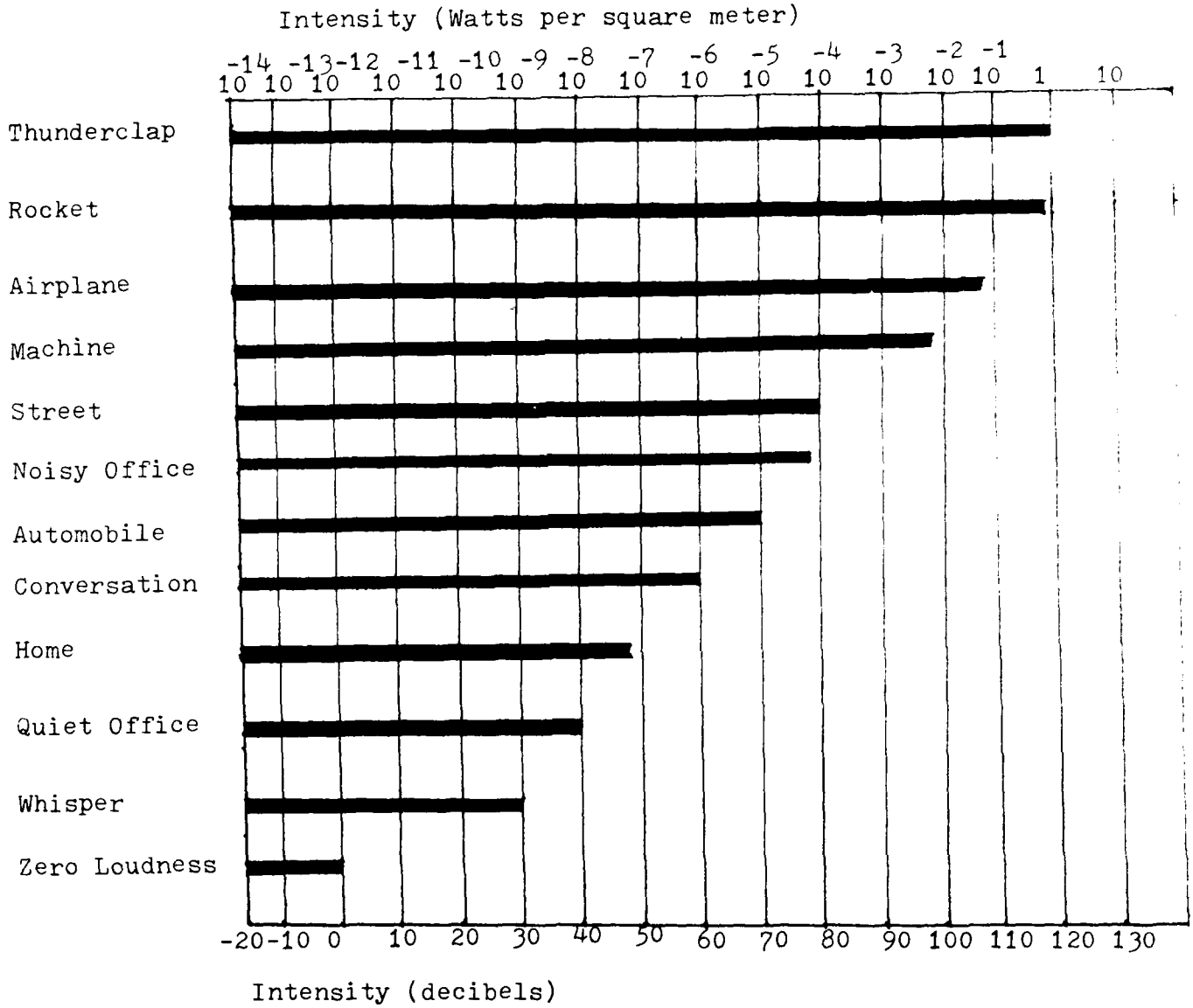


Fig 7.2 Comparative intensities of a variety of common sounds from bottom to top in order of increasing sound pressure.

7.4 References

[1] Graham Chedd," SOUND, from Communication to Noise Pollution", Doubleday & Company, Inc., New York.

[2] Wood Alexander, "Physics of Music".

CHAPTER 8
TRANSMISSION ERRORS

8.1 Introduction

Many toll-quality speech links maintain bit-error rates (BER) which are too small (less than 10^{-5}) to affect the quantizer and hence the coder performance. However, a BER of one tenth of a percent is not uncommon and for bad channels this rate could be as high as one percent. In such cases. SNR degradation may be severe unless special precautions are taken. It is important to determine the extent of SNR degradation and if possible how to minimize it.

The study outlined in this chapter is an attempt to answer the question posed above. Section 8.2 describes the method of introducing random transmission errors. This method was simulated on digital computer such that BER could be changed at run time. With the introduction of transmission errors, the effect of various parameters such as predictor order, coarseness of the quantizer and various decay constants on SNR was observed. Simulation results are discussed in Section 8.3.

8.2 Simulation of Transmission Errors:

It is assumed that transmission errors that occur in the digital channel as shown in Fig. 8.1 are random in nature. Hence in the simulation it is important to insure that errors do not occur in bursts. Similarly it is assumed that errors are made by bit reversal and not by bit addition or deletion. From the Fig. 8.1, it may appear that the effect of errors introduced in quantizer levels is similar to errors introduced in bit stream representing quantizer levels provided encoding and decoding operations are carried out correctly. However, it must be kept in mind that samples are not gained or lost if the errors are introduced in quantizer levels while they may be if errors are introduced in bit streams. The advantage of introducing errors in the quantizer levels is that the effect of transmission errors can be measured in terms of degradation of SNR so that the effect of various parameters on transmission errors can be easily evaluated. With errors in the bit stream, SNR looses its meaning since samples may no longer be synchronized. Of course, the transmission errors do affect the bit stream. The following procedure was adopted for simulation of transmission error.

1. Separate programs for PARC transmitter and receiver were written. Program asks for the bit error rate and transmitter produces a file of quantizer levels represented by integer numbers from 1 to 11. Receiver program reads quantizer levels from this file and produces file of reconstructed speech samples.
2. Randomized transmission errors were introduced by using RANDU function available in PDP 11/60 library. Care should be taken to make the seeds large enough for this function so that bursts of errors do not occur in the beginning.

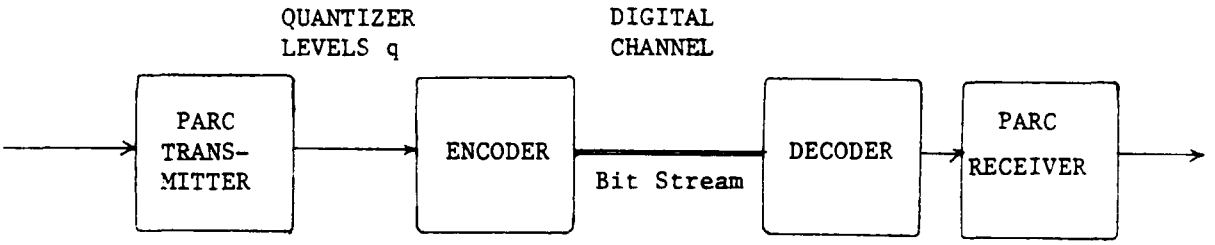


Fig. 8.1 Speech Coder

3. Original speech file and reconstructed speech file is compared and SNR is calculated. This procedure is repeated for different values of parameters.

8.3 Minimizing Transmission Error Effect

Because of error in quantizer level q(k) reconstructed reduced speech $\hat{v}(k)$ and hence reconstructed speech sample $\hat{s}(k)$ is erroneous. This makes e(k) also erroneous which in turn causes a_i's (predictor parameter) to be incorrect. The effect of $\hat{e}(k)$ on a_i's can be observed from equation 2. which is repeated here for convenience.

$$a_i(k+1) = a_i(k) + \frac{g\ \hat{v}(k-i)\ \hat{e}(k)}{[(1-\alpha)\sum_{j=o}^{\infty} \alpha^j\ \hat{v}(k-j) + RMSMIN]}$$

Effect of erroneous $\hat{e}(k)$ on updating a_i's can be minimized by increasing RMSMIN and decreasing g. This can be seen from Table 8.1 and 8.2.

Table 8.1

Male speaker, sentence 1:

Bit Error Rate 1 in 100

g	SNR without error	SNR with error
0.015	19.25 db	5.83 db
0.01	19.07 db	6.49 db

Table 8.2

Male speaker, sentence 1:

Bit Error Rate 1 in 100

RMSMIN	SNR without error	SNR with error
70	18.96 db	7.18 db
65	19.05 db	6.95 db
55	19.30 db	8.04 db
52	19.29 db	7.73 db
30	19.83 db	3.66 db

Predictor output is linear combination of past $\hat{v}$'s and is given by

$$p(k) = \sum_{i=1}^{N} a_i(k)\ \hat{v}(k-i)$$

where N is the order of predictor.

For high predictor order effect of transmission error is more since comparatively larger number of incorrect predictor coefficients and larger number of previous incorrect samples contribute to predictor output. Since a_i's and $\hat{v}$'s both are used in constructing p(k), effect of transmission errors for increase in predictor order is rather serious. It was observed that by decreasing predictor order from 8 to 4 SNR (for BER of 1 in 100) improved by 3 db.

Various decay constants such as α, exponential decay for RMS value calculation and δ, decay constant for updating predictor parameters do have effect on performance of system with transmission errors. Choice of α controls the effective interval that contributes to the Rms estimate. This interval is larger for syllabic system while it is smaller for instantaneous system. For both extremes, such as large α (syllabic) and small α (instantaneous), SNR decreased. (Ref. Table 8.3).

Decay constant δ is used to update predictor parameters to prevent the transmission errors to propagate. Larger values of δ improve the system performance with error, as can be seen from Table 8.3.

Table 8.3

Parameters	SNR with no error	SNR with 1% error rate
α = 0.97	19.28 db	6.13 db
= 0.9	18.96 db	7.18 db
= 0.8	18.89 db	5.01 db
*δ = 0.02	19.46 db	overflow
= 0.04	19.21 db	-0.78 db
= 0.06	19.18 db	1.63 db

*With all other parameters optimized for good error performance, degradation is not so severe with above values of δ.

Quantizer noise is also an important factor in affecting the performance of the system with transmission errors. Closer the quantizer levels less is the quantization noise and hence less is the effect of transmission errors. Table 8.4 shows that by taking output levels apart SNR has decreased.

Table 8.4

Symmetric quantizer.
Bit error rate 1 in 100.

Output levels	SNR without error	SNR with error
0 1.8 4.25 6.5 8 12	20.13 db	10.18 db
0 1.9 4.5 7.5 10. 12	19.42 db	7.65 db

To see the effect of various error rates and the effect of errors in different segments of speech, random errors with 1% and 0.1% error rates were added using different random sequences. The study has shown that 0.1% error rate causes little degradation while it is significant for 1% error rate. However, output speech was found to be intelligible in spite of BER of 1%. It was also noticed that if the error occurs in silence segment of speech its effect is negligible. Considering the fact that 40 to 60% of the speech is silence, effect of small BER is not significant as can be seen from Table 8.5

Table 8.5

Case 1: SNR at transmitter = 20.13 db

	Error Rate 1%	Error Rate 0.1%	
Random Sequence #1	7.13 db	15.14	(15 errors)
Random Sequence #2	8.57 db	20.00	(7 errors)
Random Sequence #3	9.78 db	14.34	(12 errors)

Case 2: SNR at transmitter = 18.63 db

	Error Rate 1%	Error Rate 0.1%
Random Sequence #1	6.99 db	16.41 db
Random Sequence #2	6.00 db	18.48 db
Random Sequence #3	8.53 db	13.72 db

8.4 Conclusion

PARC algorithm found to perform well in presence of random channel errors as high as one percent error rate. The degree of adaptation in the algorithm and error performance seems to be related. In general, higher adaptation of parameters and more complex system leads to poor performance in presence of channel errors. For example, reducing the predictor order from 8 to 4 improved the error performance significantly.

PARC employs DPCM quantizer, hence it is more tolerant to randomly occurring bit errors from perceptual point of view than systems which employ PCM quantizers (1),(2). This is because error spikes caused in the reconstruction of a PCM waveform (due to wrongly received bit) can have maximum amplitudes which are in the order of peak of input signal while corresponding spike magnitudes in DPCM decoding really related to the peak value of first difference in the input. The corsequent greater magnitude of a typical PCM error spike makes it more annoying in spite of the fact that it does not propagate in time. The effect of channel error on the synchronization has not been investigated.

8.5 References

[1] K. Y. Chang and R. W. Donaldson, "Analysis, Optimization, and Sensitivity Study of PCM Systems Operating on Noisy Communication Channels," IEEE Trans. Commun., vol. COM-20, pp. 330-350, June 1972.

[2] J. Yan and R. W. Donaldson, "Subjective Effects of Channel Transmission Errors on PCM and DPCM Voice Communications Systems," IEEE Trans. Commun., vol. COM-20, pp. 281-290, June 1972.

CHAPTER 9
FILTERING

9.1 Introduction

Subjective listening tests have indicated that the most objectionable aspect of the speech generated at the PARC receiver is its granular noise due to quantization errors. Since the spectrum of the quantization noise nq(k) will, in general, be whiter than the speech signal s(k), it appears reasonable to develop a filter for removing part of the quantization noise from $\hat{s}(k)$ and hence improving the speech.

Section 9.3 describes the pre-emphasis of speech to improve the speech quality in terms of perception. Various choices of filters are outlined in Sec. 9.3 and the results are presented in Sec. 9.4.

In the study of pre-emphasis of speech it was observed that low pass filtering does have considerable effect on entropy. This effect is discussed in Sec. 9.5 and simulation results are presented in Sec. 9.6. Design of low-pass Butterworth filter is outlined in the Appendix.

9.2 Evaluation of Pre-emphasis

For auto-correlated signals, such as speech, predictive coding [1,2] is an efficient method of encoding the signal into digital form. In predictive coders, the quantization noise depends on prediction errors; hence, efficient prediction minimizes quantization error. However, in some segments of speech, such as unvoiced speech, the degree of correlation is small and prediction is therefore poor, resulting in more quantization noise. When the amplitude of this noise is comparable to the speech signal it mars the quality of the received speech. To reduce this problem, it appeared that some sort of pre-emphasis of speech would be helpful.

The basic concept of a pre-emphasis filter as shown in Fig. 9.1 is to spread energy in the input signal over the full bandwidth of the processor. Since most of the energy in speech is in the lower end of the spectrum, the filter is a high-pass filter; and hence, the de-emphasis filter is a low pass filter. As pre-emphasis filter is a high frequency filter, the unvoiced segment of speech gets emphasized. However, the filter design is such that overall energy gain for typical phonetically balanced sentence is approximately to unity.

From the figure, it is clear that pitch extraction is to be carried out on high pass filtered speech to get β's and T's. However, it was observed that it makes little difference if the β's and T's are obtained by pitch extraction on original speech.

It might be possible to manipulate the block diagram of PARC by moving filters inside to get the equivalent system. This is of no immediate interest and hence not covered here. However, PARC algorithm

could be modified to include adaptive noise spectral shaping as proposed by Makhoul & Berouti [3].

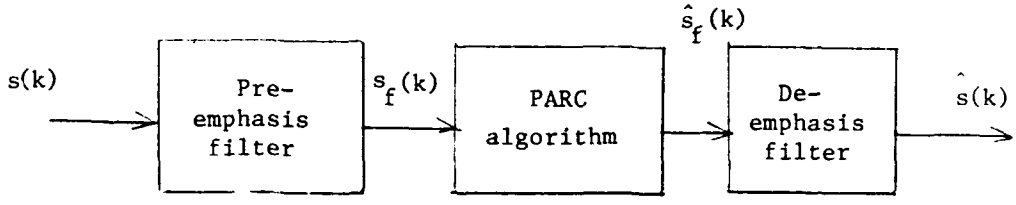


Fig. 9.1 Pre and De-emphasis Filters with PARC Algorithm

9.3 Filter Selection

As mentioned earlier, the pre-emphasis filter is a high-pass filter and de-emphasis is just the inverse of pre-emphasis filter.

In order to minimize complexity, the order of filter is kept small. It was decided to use a 3rd-order filter with general form

$$s_f(k) = \frac{1}{K} s(k) + \frac{a}{K} s(k-1) + \frac{b}{K} s(k-2) + \frac{c}{K} s(k-3) \qquad (9.1)$$

The Z transform of this filter is

$$H(z) = \frac{1 + aZ^{-1} + bZ^{-2} + cZ^{-3}}{K} \qquad (9.2)$$

Then, using the fact that Z is just $e^{j\omega T}$ where T is the sampling interval the system function becomes

$$H(j\omega) = \frac{1 + ae^{-j\omega T} + be^{-j2\omega T} + C\, e^{-j3\omega T}}{K} \qquad (9.3)$$

and the magnitude of the system function is

$$|H(e^{j\omega t})|^2 = 1 + a^2 + b^2 + c^2 + [2a + 2b\ (a + c)] \cos \omega T + 2(b + ac) \cos 2\omega T + 2c \cos 3\omega T \Big/ K \qquad (9.4)$$

The four filters whose performance will be described in details in the next section are shown in Table 9.1.

Table 9.1
Filter Parameters

	K	a	b	c
filter 1	0.5113	-1.182	0.677	-0.140
filter 2	0.5718	-0.888	0.486	-0.095
filter 3	0.6718	-0.626	0.347	-0.064
filter 4	0.7311	-0.508	0.219	-0.053

The frequency response of these filters is shown in Fig. 9.2 for a sampling rate of 6.4 kss.

The de-emphasis filter is just the inverse of the pre-emphasis filter. Therefore, for filter of Eq. (9.3) the inverse is

$$\hat{s}(k) = K\, s_f(k) - a\hat{s}(k-1) - b\hat{s}(k-2) = c\hat{s}(k-3) \qquad (9.5)$$

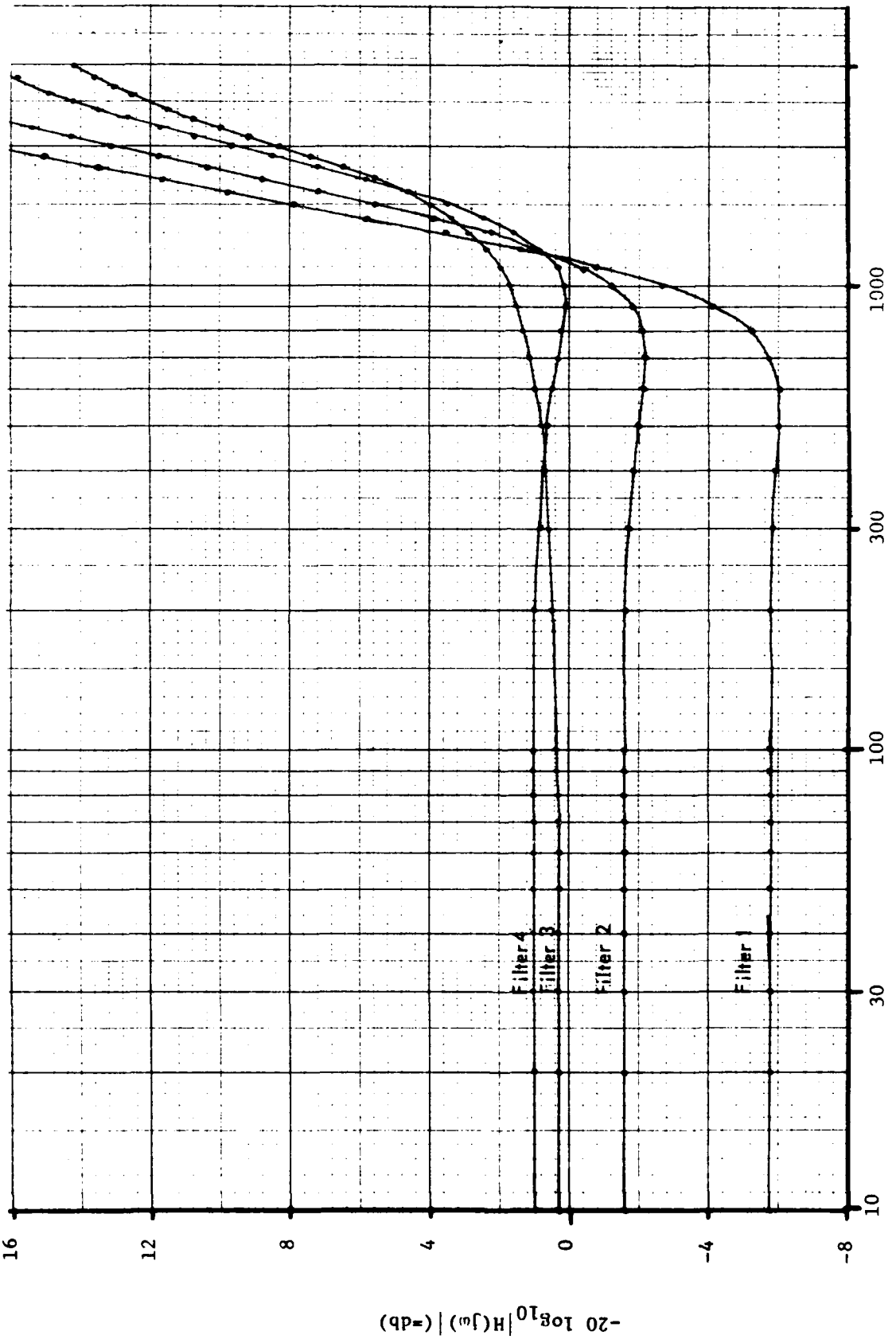


Fig. 9.2 Frequency Response of Pre-emphasis Filters

9.4 Results

The four pre-emphasis filters shown in Fig. 9.2 were evaluated on a PARC operating in the 9.6 kbs mode. Sentence 1, "Cats and Dogs each hate the other" was used for the simulation. Following parameters were computed to evaluate the performance of the filters.

$$\text{SNR(PARC)} = \frac{\sum s_f^2(k)}{\sum (s_f(k) - \hat{s}_f(k))^2} \tag{9.6}$$

$$\text{SNR(overall)} = \frac{\sum s^2(k)}{\sum (s(k) - \hat{s}(k))^2} \tag{9.7}$$

$$\text{SEGSNR} = \frac{\sum_{i=1}^{n} \text{SNR}_i}{n} \tag{9.8}$$

where n - number of blocks of block length 120 samples in this case.

$$\text{SER (Signal Energy Reduction)} = -10 \log \frac{\sum v_f^2(k)}{\sum s_f^2(k)} \tag{9.9}$$

where $v_f(k) = s_f(k) - \beta\, \hat{s}_f(k-T)$

It was noticed that by pre-emphasizing speech, the output speech is perceptually better than without pre-emphasis. However, there is an increase in entropy value. This is due to the fact that the increased amplitude of high frequency speech generates more upper levels of quantizer thus generating more bits. All results are reported in Table 9.2.

Table 9.2

Signal to Noise Ratios
with Various Pre-emphasis Filters

	Product of Pre & De-emphasis Filter Gain	SNR PARC	SER	SNR Whole System	SEGSNR	Entropy H bits/sec
no filters	-	18.54 db	5.43 db	18.54 db	11.02 db	1.50
filter 1	1.003	15.63 db	2.71 db	16.73 db	9.68 db	1.97
filter 2	0.994	16.44 db	3.58 db	18.64 db	11.02 db	1.88
filter 3	0.999	17.41 db	4.44 db	19.36 db	11.65 db	1.74
filter 4	0.997	17.69 db	4.73 db	19.20 db	11.55 db	1.69

9.5 Low-pass Filtering vs. Entropy

For the 9.6 Kbs transmission rate and 6.4 KHz sampling frequency, the number of bits per sample is 1.5. Transmission of parameters such as β and T take a few bits per sample. Therefore, the entropy in the simulation must be maintained at the value less than 1.5. In the previous section, it was observed that pre-emphasis makes the output speech perceptually better; however, it also increases entropy which is unacceptable. To overcome this problem one could use coarser quantization to make the entropy small to begin with and then employ pre- and de-emphasis filters. Unfortunately, the improvement in speech quality due to pre-emphasis operation is not significant enough to consider the above approach.

Another method for achieving good speech quality while controlling the bit rate would be to select parameters such that the speech quality is excellent disregarding the increase in entropy and using the buffer control to check the bit rate. How this buffer control works is discussed in details in the next chapter. The use of a filtering operation to control the bit rate is discussed here.

If a low pass filter is used instead of high pass filter as a pre-emphasis filter, energy reduction is improved and as a result entropy drops. Therefore, the buffer will fill at a slower rate and buffer control would be used infrequently; consequently, there would be less degradation caused by the use of buffer control. However, low pass filtering with bandwidth less than 3200 Hz causes some loss of speech naturalness. It was noticed that during high energy, voiced segment, of speech that the bit rate is higher and hence the buffer fills faster. If the bit rate is brought down by employing low-pass filtering after buffer content is greater than a particular threshold, a double purpose is served. One, the buffer filling

operation is slowed down thus avoiding or delaying the drastic buffer control operation. Second, high frequency componets in low energy unvoiced segments of speech are not filtered out since filter is in operation only after particular threshold thus preserving natural quality. This threshold was found by plotting the buffer content against time for a typical sentence and noting the value of buffer content for voiced speech. For the bit buffer in the FORTRAN simulation of the PARC algorithm, 500 bits appeared to be reasonable a threshold value.

As the cut-off frequency of low-pass filter is decreased, the performance of the predictor improves thus decreasing bit rate further. Thus, the filter cut-off frequency can be decresed depending how full the buffer gets. Hence the pre-emphasis filter is now an adaptive low-pass filter, adaptation of the cut-off frequency depending on the buffer contents.

The low-pass filter used is a simple 3rd order Butterworth filter. Its design and frequency plots are given in an appendix to still chapter.

9.6 Results and Conclusion

The FORTRAN simulation of PARC was modified to include the adaptive low-pass filter concept. To insure that different filters are not employed for every sample of speech when the buffer contents are close to the threshold, a hysterisis structure was utilized. Once a particular filter is selected, it employed in the algorithm for next block of 100 samples. The buffer content is compared with the thresholds only after the block of speech samples is processed. This is shown in the flow chart in Fig. 9.3. The simulation was carried out for Sentence 1: male speaker. The effects of low-pass filtering on entropy and signal energy reduction are tabulated in Table 9.5 while effects on buffer content and bit rates are reported in Table 9.6.

Table 9.5

Effects of Low pass Filtering

	SNR	SER	Entropy H
no filter	14.85 db	4.1 db	1.37 bits/sample
LPF with cut-off 1400 Hz	16.43 db	5.1 db	1.95 bits/sample

Table 9.6

Effect of LPF on Bit Rate

	Sample Number	Change in Buffer Content	Increase Bit/ Sample	# Bit Rate
no filter	300 - 1000	800	1.14	2.49
LFP with Cut-off 1400 Hz	300 - 1000	680	0.97	2.32

*Bit rate is obtained by adding 1.35 bits/sample to the bit/sample increase in buffer. This is because on an average of 1.35 bits/ samples are transmitted on digital channel.

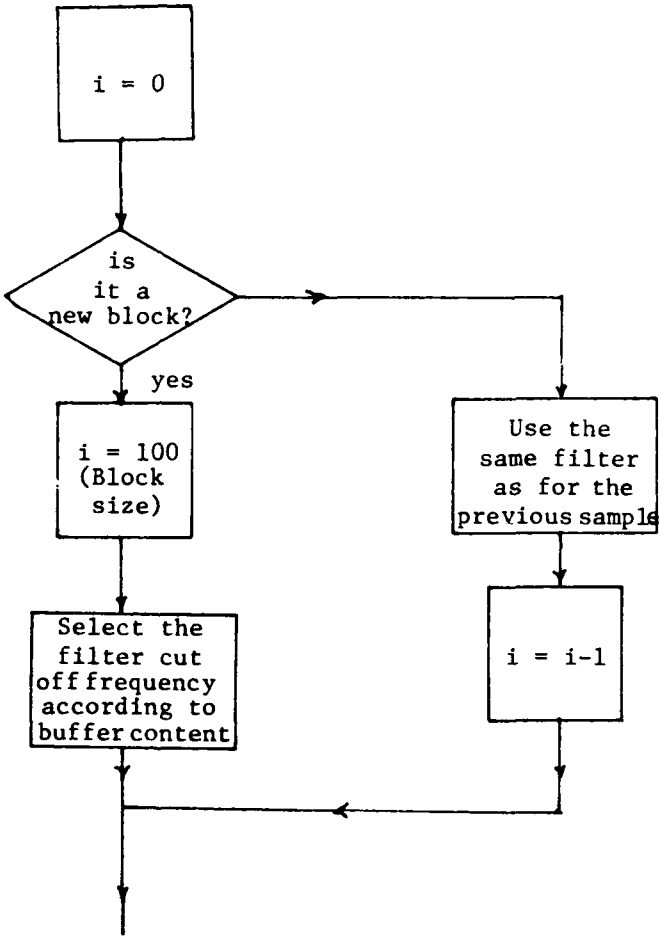


Fig. 9.3 Hysterisis Structure of Adaptive Low Pass Filter

The results in Table 9.5 and 9.6 indicate that Low-pass filtering can be used to control bit rate. In the real-time simulation on the MAP, the structure of programming was such that a varying number of speech samples would be processed each time to get exact predetermined number of bits. This could cause double or triple filtering of the same speech samples. This situation occurs in voiced segments of speech.

The all-pole 3rd order filter could not be used since double and triple filtering causes frequency response to have peaks at the 3 db frequency. This is undesirable and hence there is a need to find new methods of employing adaptive low filtering for buffer control.

It was observed that multiple filtering reduces the 3 db frequency. As mentioned earlier more bits are generated in voiced speech and filter cut-off frequency has to be reduced to cut down the bit rate. Since multiple filter case happens in voiced speech and since it is the region where low-pass cut-off frequency needs to be decreased, the design of single filter would be enough. Thus, there would be no need to change the filter as buffer gets closer to being full. This happens automatically by multiple filtering when buffer gets closer to being full. The filter which gives required change in 3 db frequency upon repetition was designed and design and frequency response is outlined in the appendix.

9.7 References

[1] P. Elias, "Predictive Coding," IRE Trans. Inform. Theory, vol. IT-1, pp. 16-33, Mar. 1955.

[2] B. S. Atal and M. R. Schroeder, "Adaptive Predictive Coding of Speech Signals," Bell Syst. Tech. J., vol. 49, pp. 1973-1986, Oct. 1970.

[3] Makhoul, John and Michael Berouti, "Adaptive Noise Spectral Shaping & Entropy Coding in Predictive Coding of Speech", IEEE Trans. on Acoust. Speech Signal Processing, vol. ASSP-27, No. 1, Feb. 1979, pp. 63-73.

[4] D. Childers and A. Durling, "Digital Filtering and Signal Processing", West Publishing Company.

9.A Appendix: Derivation of Filters

9.A.1 Butterworth (Maximally flat) Filters

A Butterworth filter is designed to be maximally flat at the origin of the magnitude of the frequency response, i.e., the filter is forced to have as many zero derivatives at the origin of the magnitude response as possible.

The normalized squared magnitude response of the Butterworth filters is

$$|H(\omega)|^2 = \frac{1}{1 + \omega_p^{2n}} \tag{9.8}$$

where $\omega_p = \omega/\omega_c$ is the normalized frequency for an nth order and ω_c is the desired 3 db cut-off frequency of the nth order filter.

In the design of the desired filter frequency response the poles of the transfer function H(s) are needed, then

$$H(s)H(-s) = \frac{1}{1 + (-s^2)^n} = \begin{cases} 1/(1 + s^{2n}) & n \text{ even} \\ 1/(1 - s^{2n}) & n \text{ odd} \end{cases} \tag{9.9}$$

Thus, the 2n roots of ± 1 are desired depending on the oddness or evenness of the order of the desired filter. Consider the third-order Butterworth filter; for n=3, the poles are

$$s_0 = \omega_c \angle 0^\circ \qquad s \pm 1 = \omega_c \angle \pm 60^\circ \qquad s \pm 2 = \omega_c \angle \pm 120^\circ$$

$$s_3 = \omega_c \quad 180^\circ$$

These poles are plotted in Fig. 9.4.

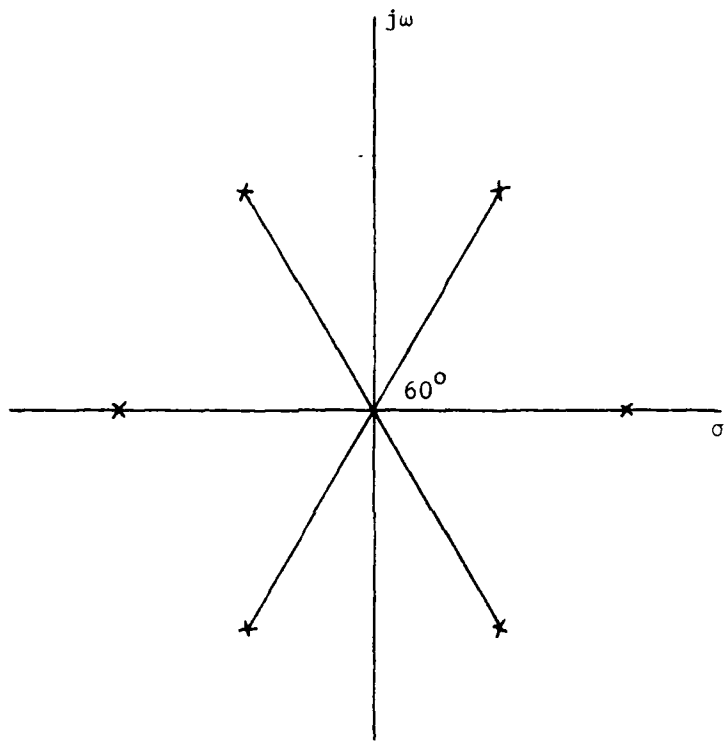


Fig. 9.4 Pole Location of 3rd Order Butterworth Filter

Since the poles of the magnitude squared transfer function are symmetrically placed, the poles which fall in the left half plane are assigned to H(s) for physical realizability.

Thus

$$H(s) = \frac{1}{\prod_{i=1}^{3} (s - s_i)} \tag{9.10}$$

where s_i are the left half plane poles of the magnitude squared transfer function. In the third order case

$$s_1 = (-\frac{1}{2} - j\frac{\sqrt{3}}{2})\omega_c$$

$$s_2 = (-\frac{1}{2} + j\frac{\sqrt{3}}{2})\omega_c$$

$$s_3 = -\omega_c$$

The equivalent transfer function for the digital filter becomes

$$H(z) = \frac{Kz^3}{(z - p_1)(z - p_2)(z-p_3)} \tag{9.11}$$

where

$$p_1 = e^{s_1 T} = \exp\,[(-\frac{1}{2} - j\frac{\sqrt{3}}{2})\omega_c T]$$

$$p_2 = e^{s_2 T} = \exp\,[(-\frac{1}{2} + j\frac{\sqrt{3}}{2})\omega_c T]$$

$$p_3 = e^{s_3 T} = \exp\,[-\omega_c T]$$

H(z) can also be written as

$$H(z) = \frac{K}{1 + az^{-1} + bz^{-2} + cz^{-3}} \qquad (9.12)$$

Comparing (9.11) and (9.12)

$$a = -e^{\frac{-\pi f_c}{f_s}} \left[e^{\frac{-\pi f_c}{f_s}} + 2 \cos\left(\frac{\sqrt{3}\pi f_c}{f_s}\right) \right]$$

$$b = -e^{\frac{-2\pi f_c}{f_s}} \left[1 + 2 e^{\frac{-\pi f_c}{f_s}} \cos\left(\frac{\sqrt{3}\pi f_c}{f_s}\right) \right]$$

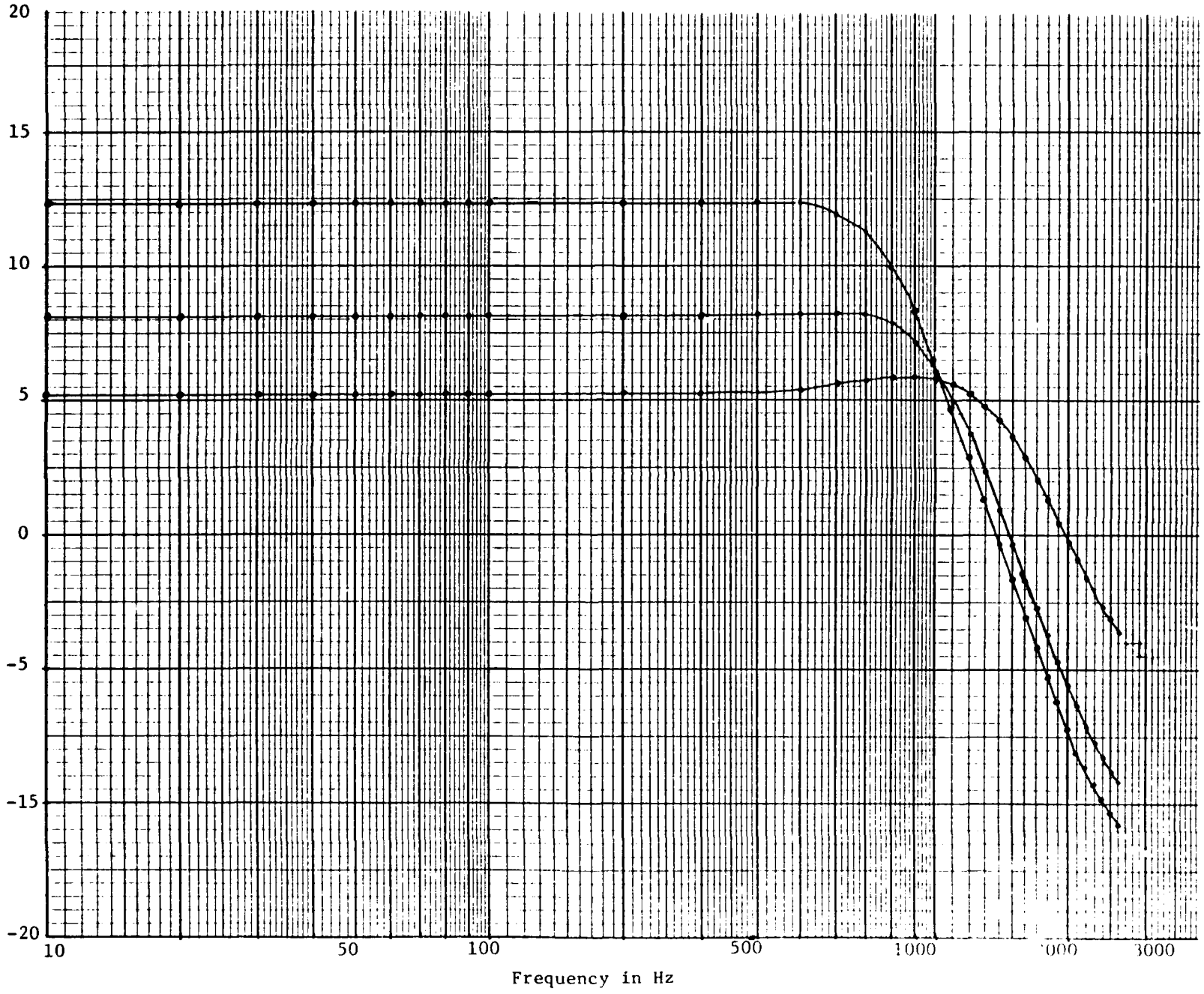
$$c = -e^{\frac{-4\pi f_c}{f_s}}$$

Where fc - Cut off frequency in Hz.

f_s - Sampling frequency in Hz.

Frequency response of various Butterworth filters is plotted in Fig. 9.5.

Fig. 9.5 Frequency Response of 3rd Order Butterworth Filter



9.A.2 Design of the low-pass filter used in the algorithm

As mentioned in the earlier sections of this chapter, low pass filtering was seen as a method of softly degrading the speech to avoid overflowing the speech buffer. This low pass filter had to be designed, however, to operate in the MAP.

The first design of the necessary digital filter was done using the impulse invariant transformation on a third order Butterworth filter. Unfortunately, the resulting digital filter had a relatively large amount of ripple which was not desirable, especially because multiple filtering was desired. To assure monotonicity, then, the digital filter was re-designed, using the conformal bilinear transform. Also, it appeared that for ease of implementation in the MAP that the digital filter has only one zero in the z-plane. Thus, the general form of the transfer function of the digital filter was

$$H(z) = \frac{\omega_{CA} + \omega_{CA} z^{-1}}{(\omega_{CA} + 1) + (\omega_{CA} - 1) z^{-1}} \qquad (9.13)$$

where

$$\omega_{CA} = \tan\left(\frac{\pi}{6400} f_c\right),$$

$$f_C = \text{filter cutoff frequency (Hz)}.$$

The frequency response of such a filter with an 1800 Hz cutoff is shown in Fig. 9.6. The frequency response for double filtering is also shown. and it can be seen that that cutoff frequency is about 1350 Hz.

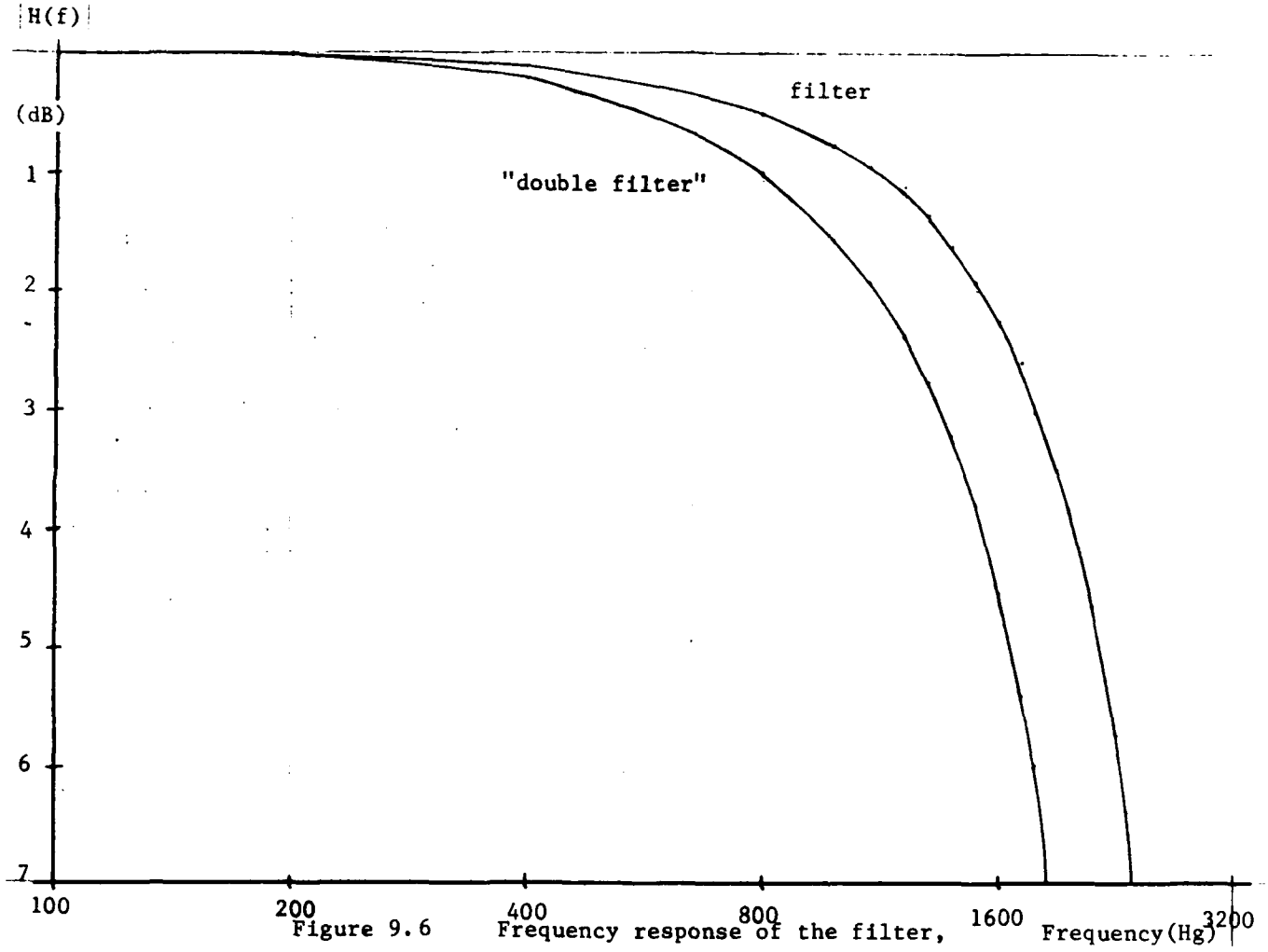


Figure 9.6 Frequency response of the filter, and the "double filter".

CHAPTER 10
BUFFER CONTROL

10.1 Introduction

Even with the use of adaptive low pass filtering, as described in the previous chapter, to decrease bit-rate generation when the speech buffer approaches overflow, there still remains the ultimate problem of deciding what to do to prevent the buffer from overflowing. Similarly, there is also the problem of deciding what to do to prevent the buffer from underflowing. These two problems are considered in this chapter on buffer control.

10.2 Overflow Control

To prevent buffer overflow, a method had to be found to sharply limit the bit generation rate occasionally which did not cause an unreasonable amount of distortion. Because the quantizer employs feedback and is backward adaptive, it is possible to obtain buffer control by denying the use of certain quantizer levels (or by selectively permitting the use of additional quantizer levels), or by varying the decision thresholds for the quantizer levels. A number of simulation runs were made of both of these techniques without a great deal of success. It appeared that dropping or adding levels was too crude to be an effective control. It was found that either produced a very small change in bit rate generation, or a very pronounced change in bit rate generation. Varying the decision threshold did not appear to be very useful, either, because it typically caused too much distortion.

As a result of these investigations, the problem was approached again from a different, angle. Analysis of simulation results showed that the speech buffer was most prone to overflow during instances of voiced speech. This suggested that pitched repetition might be a useful solution.

Pitched repetition relies on the large amount of correlations between pitch periods of voiced speech. In pitched repetition, samples are generated by duplicating the samples from one pitch period earlier. Due to the large amount of correlation, pitched repetition can typically be carried out for short periods of time during voiced speech, without greatly affecting the subjective quality of speech.

Details of the implementation of pitched repetition are given in Chapter 2.

10.3 Underflow Control

It is just as important to prevent the transmitter sample buffer from underflowing as it is to prevent it from overflowing. This is because when the transmitter sample buffer underflows, the receiver sample buffer overflows. Thus, some method was needed to prevent the transmitter sample buffer from underflowing.

A relatively easy solution to this problem was found by the use of "null" quantizer levels. This technique involves the transmission of a specified bit pattern, just like a normal quantizer level, except that it causes nothing to happen and is discarded at the receiver, with nothing being placed in the receiver sample buffer. Null quantizer levels are used as necessary, then, to prevent the transmitter sample buffer from underflowing.

10.4 Special Considerations at the Receiver

In the error-free condition, it is possible to control both the transmitter and receiver sample buffers by controlling just the transmitter sample buffer. In the presence of errors, however, this is no longer the case. For example, due to an error, it would be possible for the receiver sample buffer to underflow without the transmitter sample buffer overflowing. Some simple rules were developed to handle this situation and to resynchronize the buffers. If the receiver sample buffer overflows, the most recent sample is discarded, since it probably represents silence or near silence. This seems to cause the least distortion, and resynchronizes the buffers. If the receiver sample buffer underflows, a quantizer level "1" is inserted. This again causes a minimum of distortion and resynchronizes the buffers.

10.5 Conclusions and Suggestions for Further Research

It is necessary to provide buffer control to prevent potentially catastrophic conditions due to overflow or underflow. Some relatively simple, but effective, strategies for buffer control have been developed to this end.

There is, of course, room for improvement. For example, the null quantizer levels could also be used to force resynchronization of the transmitter and receiver sample buffers. Even more basic questions exist about the problem of buffer control itself, because it would seem that the quantization system is not as efficient as it could be if it regularly runs into overflow and underflow. A related question is why pitched repetition, which takes few bits to transmit, is so effective at a time when the quantizer is operating at a high bit generation rate.

CHAPTER 11
SOURCE AND ERROR CONTROL CODING

11.1. Introduction

Source and error control coding are the interface between the internal variables of the system and the communications channel. The noiseless source coder performs the first step in generating the bits to be transmitted. Its goal is to try to convey all the necessary information using a minimum of bits. The error control coding is then used to increase the probability that these bits will be received without error. Due to differences in quantity and importance, though, the quantizer levels and the side information are handled in different ways by the source and error control coders.

11.2. Source Coding

11.2.1. Quantizer Levels

In most common quantization systems, a very simple source coding procedure is used: typically there are 2^N quantizer levels used by the system, and each quantizer level is coded as a N-bit binary number. In contrast, PARC uses a more sophisticated source coding procedure in order to increase performance. The source code used to encode the 11 quantizer levels and the "null" level use a variable number of bits to represent the levels, with fewer bits being used for the more common levels. In this way, the quantizer level information can be conveyed very efficiently.

The first attempts at designing a simple variable length source code, however, proved frustrating. Simulations showed that a simple variable length source code tended to cause the sample buffer to fill rapidly during segments of voiced speech. Analysis of the situation showed that the problems appeared to stem from the fact that the quantizer levels were not stationary or independent. Instead, analysis showed that the quantizer levels were better represented by a model where the levels were generated by switching between two sources, one representing the quantizer behavior during voiced segments, and another representing the quantizer behavior during unvoiced and silent segments.

In order to take advantage of this phenomenon, then, a new variable length source code was developed. This new source code was what is known as an overfull source code. Overfull codes are characterized by an ability to encode some sequence in more than one way. For example, in the final source code, a sequence of 14 level-1's could be encoded in either of two ways. This

redundancy would appear, at first glance, to decrease the efficiency of the code, but it, in fact, increases the efficiency of the code. In particular, this redundancy is what allows the source code to perform well with a bimodal source. This was accomplished by designing most of the code using the high-entropy (voiced) statistics, and then adding the long-run codeword based on the low-entropy statistics. In this way, the overfull code performs better than a non-overfull code could.

11.2.2. Side Information

Besides encoding the quantizer levels, the source coder must also encode the side information used in the PARC system. This information consists primarily of the pitch extraction coefficient β and the pitch period T. (Pitched repetition is also signaled by the use of a false β.) This encoding is performed for every frame of samples.

The encoding used is fairly straightforward. The pitch period can only be one of 64 possible integers between 20 and 83, so that it can be represented exactly by 6 bits. The encoding used for the pitch extraction coefficient β, however, is slightly more complicated. The first complication is that the value of β must be quantized because it is a real number. It was determined by simulation, though, that the system was relatively insensitive to the quantization of β, and that using 97 quantization levels, evenly distributed between -2 and 2, appeared to have a negligible effect. The other complication was the signaling of pitched repitition through the use of a false β. The signaling itself could be handled easily by simply assigning it an unused β value. It was felt, however, that it was important that this signal not be mistaken. The encoding used for β, then, used 7 bits, with the all zero pattern reserved for pitched repetition. Error suppression was then provided by not assigning any β quantizer levels to the patterns containing one or two 1's. This left 99 patterns for β quantizer levels, while protecting the pitched repetition signal from single bit errors.

11.3. Error Control Coding

In order to maintain system performance when using a communications channel with a relatively high bit error rate, error control coding is provided for each block of bits. There were several constraints, though, which dictated what kind of error control coding could be used. The blocks were required to be about 200 bits long by the pitch information. The block length also was constrained by synchronization requirements; it was also constrained by the lengths which simple coding schemes require. All of the constraints were satisfied by performing the coding over partial blocks, rather than an entire block at a time. The 189 bit block is divided into three 63 bit frames, so that a single-error-correcting (57, 63) Hamming code may be used. As a result, up to 3 bit errors per block can be corrected, greatly improving the performance of the system in a severe environment.

11.4. Conclusions and Suggestions for Further Research

The source and error control coding described in this chapter allows the system to perform efficiently and cope with severe communications environments. The schemes described were rather simple, as appeared to be required for this implementation. There are, of course, more sophisticated methods which could be studied which might improve performance even further. A major question is how to develop quantization schemes which allocate bits efficiently according to subjective criteria. Another major issue is how to design error protection systems which perform well over a large range of bit error rates.

APPENDIX A
FORTRAN SIMULATION OF ALGORITHM

This appendix presents a listing of the FORTRAN simulation of PARC algorithm. This simulation differs from the real-time algorithm in two ways. First, this algorithm operates on a block containing a fixed number of samples rather than a fixed number of bits. Second, the algorithm described here does not have the adaptive filtering of the input sample sequence.

```
;
;       COMMAND FILE TO BUILD THE TASK FOR PARC SIMULATION
;
REPEAT/CP/FP,REPEAT/NOSP=REPEAT,PITCH,INSTR5,LPPROC,MODB,INLOO4,GROW4,
BUFCON,RANKS4,PRUNE5,LEFST,MFILE,SNRVAL,SAVE,OUTBUF,INEND5
/
ACTFIL=7
UNITS=1#
//
```

```
FORTRAN IV-PLUS V#2-51          15:47:32    15-APR-8#          PAGE 1
REPEAT.FTN      /TR:BLOCKS/WR

###1            PROGRAM REPEAT
        C
        C       PURPOSE:  TO SIMULATE THE REPEAT ALGORITHM.
        C
        C       V.R     DATE            NAME            COMMENTS
        C       #.1     16-JUL-79       J.M. KRESSE     EXISTING SOFTWARE
        C       1.#     17-JUL-79       J.M. KRESSE     BACKED UP
        C       1.1     2#-JUL-79       J.M. KRESSE     SPLIT OFF LPPROC, MODB
        C       1.2     22-JUL-79       J.M. KRESSE     OUTPUT BUFFER COUNTER
        C       1.1#    31-OCT-79       J.M. KRESSE     REMAINING SAMPLES BUG
        C
        C       THIS IS A SPEECH CODING ALGORITHM FOR DIGITAL TRANSMISION OF
        C       SPEECH AT A RATE OF 96## BITS PER SECOND.THE ALGORITHM
        C       COMBINES PITCH EXTRACTION LOOP,PITCH COMPENSATING ADAPTIVE
        C       QUANTIZER,SEQUENTIALLY ADAPTIVE PREDICTOR,MULTIPATH TREE
        C       SEARCHING AND ADAPTIVE SOURCE CODING.ADAPTIVE SOURCE CODING
        C       PART OF THE ALGORITHM WILL BE ADDED TO THIS LATER.
        C
        C       SPEECH SIGNAL IS HIGHLY REDUNDANT.THIS REDUNDANCY IS REMOVED
        C       BY EXTRACTING PITCH BLOCK BY BLOCK AND PITCH REDUCED SIGNAL
        C       IS FORMED.A SEQUENTIALLY ADAPTIVE PREDICTOR USING BACKWARD
        C       ADAPTATION,IS USED TO FORM AN ESTIMATE OF PITCH REDUCED SIGNAL.
        C       THE ERROR IN THIS ESTIMATE IS QUANTIZED BY PITCH COMPENSATING
        C       ADAPTIVE QUANTIZER.THE QUANTIZER HOWEVER USES THE MULTIPATH
        C       TREE SEARCHING ALGORITHM TO DETERMINE ITS OUTPUT.THE RESULTING
        C       QUANTIZED OUTPUT IS CODED USING AN ADAPTIVE SOURCE CODING
        C       PROCEDURE.SOURCE CODE ALSO PERMITS TRANSMISSION OF PITCH
        C       INFORMATION AND SYNCHRONIZING SIGNALS.
        C
        C       FOLLOWING IS THE LIST OF VARIABLES.
        C       S               -ARRAY OF ORIGINAL SPEECH SAMPLES.
        C       V               -ARRAY OF REDUCED SPEECH SAMPLES.
        C       VHAT            -ARRAY OF ESTIMATED PITCH REDUCED SPEECH.
        C       SHAT            -ARRAY OF RECONSTRUCTED SPEECH SAMPLES.
        C       T               -PITCH PERIOD.
        C       BETA            -CORRELATION COEFFICIENTS.
        C       KBLK            -BLOCK SIZE.
        C       NBLK            -NUMBER OF BLOCKS.
        C       L               -DEPTH OF TREE IN TREE SEARCHING ALGORITHM.
        C
        C
```

```
FORTRAN IV-PLUS V02-51          15:47:32     15-APR-80          PAGE 2
REPEAT.FTN      /TR:BLOCKS/WR

      C         PARAMETER BUFFL=16,IBUF1=256,IBUF2=512,KBLK=150                  1-1.7
0002            PARAMETER BUFFL=16,IBUF1=256,IBUF2=512,KBLK=100                  11.7
0003            PARAMETER LUNBUF=7,LUNPRT=6,LUNSH=2,LUNTI=5,MXBLCK=200           11.2
0004            PARAMETER NAMLEN=33,NO=.FALSE.,NUL=0,YES=.TRUE.                  11.2
      C         PARAMETER LUNPRT=6,LUNSH=2,MXBLCK=200                            1-1.2
0005            BYTE FNAME(NAMLEN),FNAME1(NAMLEN),FNAME2(NAMLEN),FNAME3(NAMLEN)  11.2
0006            BYTE FNAME4(NAMLEN),H(40),ICOMM(40)                              11.2
      C         BYTE FNAME(33),FNAME1(33),FNAME2(33),FNAME3(33),H(40),ICOMM(40)  1-1.2
0007            BYTE IDATE(10),ITIME(9),QPLOT
0008            INTEGER NCHAR,PREFIX                                             11.2
0009            INTEGER Q,S,T(MXBLCK)                                            11.1
      C         INTEGER Q,S,T(200)                                               1-1.1
0010            LOGICAL BUFFIL                                                   11.2
0011            REAL BUFCNT                                                      11.2
0012            DIMENSION BETA(MXBLCK),SHAT(IBUF1),V(IBUF1),VHAT(IBUF1)          11.1
      C         DIMENSION BETA(200),SHAT(IBUF1),V(IBUF1),VHAT(IBUF1)             1-1.1
0013            COMMON S(IBUF2)
0014            COMMON /ICNT/ ICNT
0015            COMMON /MBUFF/ MBUFF(BUFFL)
0016            COMMON /QPLOT/ QPLOT
0017            MODB1(INDX1)=IAND(INDX1-1,IBUF1-1)+1
0018            MODB2(INDX2)=IAND(INDX2-1,IBUF2-1)+1
      C
      C         ASK FOR THE FILENAMES, COMMENTS
      C
0019            WRITE(5,510)
0020  510       FORMAT(' ENTER THE INPUT FILENAME:')
0021            READ(5,520)FNAME
0022  520       FORMAT(33A1)
0023            WRITE(5,530)
0024  530       FORMAT(' ENTER THE PARAMETER POINTER FILENAME:')
0025            READ(5,540)FNAME1
0026  540       FORMAT(33A1)
0027            WRITE(5,550)
0028  550       FORMAT(' ENTER THE OUTPUT FILENAME:')
0029            READ(5,560)FNAME2
0030  560       FORMAT(33A1)
0031            WRITE(5,570)
0032  570       FORMAT(' ENTER THE OUTPUT HEADER COMMENT:')
0033            READ(5,580)ICOMM
0034  580       FORMAT(40A1)
      C
      C         DO UNTIL Y OR N
      C
0035  590       CONTINUE
0036                  WRITE(5,591)
0037  591             FORMAT(' DO YOU WANT AN EFFECTIVE M PLOT? (Y OR N)')
0038                  READ(5,592)QPLOT
0039  592             FORMAT(A1)
0040                  IF(.NOT.((QPLOT.EQ.'Y').OR.(QPLOT.EQ.'N')))GO TO 590
0041            CONTINUE
0042            IF(QPLOT.NE.'Y')GO TO 594
0043                  WRITE(5,593)
0044  593             FORMAT(' ENTER THE EFFECTIVE M FILENAME:')
0045                  READ(5,560)FNAME3
```

```
0046    594     CONTINUE
0047            WRITE(LUNTI,595)                                        11.2
0048    595     FORMAT(' IF YOU WANT A FILE OF BUFFER COUNTS, ENTER THE',  11.2
                1 ' FILENAME.'/                                         11.2
                2 ' OTHERWISE, HIT RETURN.')                            11.2
0049            READ(LUNTI,596)NCHAR,FNAME4                             11.2
0050    596     FORMAT(Q,<NAMLEN>A1)                                    11.2
0051            IF(NCHAR.EQ.0)GO TO 597                                 11.2
0052                    BUFFIL=YES                                      11.2
0053                    FNAME4(NAMLEN)=NUL                              11.2
0054                    OPEN(UNIT=LUNBUF,NAME=FNAME4,TYPE='NEW',        11.2
                1        CARRIAGECONTROL='LIST')                        11.2
0055            GO TO 598                                               11.2
0056    597     CONTINUE                                                11.2
0057                    BUFFIL=NO                                       11.2
0058            GO TO 598                                               11.2
0059    598     CONTINUE                                                11.2
        C
        C       INSERT END-OF-STRING MARK.
        C
0060            FNAME(33)=0
0061            FNAME1(33)=0
0062            FNAME2(33)=0
0063            FNAME3(33)=0
```

```
FORTRAN IV-PLUS V02-51          15:47:32    15-APR-80          PAGE 4
REPEAT.FTN      /TR:BLOCKS/WR


        C
        C
        C
        C       OPEN THE INPUT FILE.
        C
0064            OPEN(UNIT=1,TYPE='OLD',READONLY,NAME=FNAME,SHARED)
        C
        C
        C       READ THE HEADER ON SPEECH FILE.
        C
0065            READ(1,10)NSENT,IRATE,NSAMP,IUPPR,ILOWR,NTERMS,H
0066       10   FORMAT(6I5,10X,40A1)
        C
        C       PRINT THE HEADER.
        C
0067            CALL DATE(IDATE)
0068            CALL TIME(ITIME)
0069            WRITE(6,11)IDATE,ITIME
0070       11   FORMAT('1DATE: ',10A1,' TIME: ',9A1)
0071            WRITE(6,12)
0072       12   FORMAT('0THE SPEECH FILE HEADER:')
0073            WRITE(6,20)NSENT,IRATE,NSAMP,IUPPR,ILOWR,NTERMS,H
0074       20   FORMAT(1X,6I5,10X,40A1)
        C
        C
        C
0075            WRITE(6,70)KBLK,NSAMP
        C
0076       70   FORMAT('0FRAME SIZE=',I4/' TOTAL SAMPLES=',I6)
        C
0077            WRITE(5,71)
0078       71   FORMAT('0ENTER THE VALUE OF THE MULTIPLYING FACTOR ALPHA:')
0079            READ(5,72)ALPHA
0080       72   FORMAT(G15.7)
0081            WRITE(6,73)ALPHA
0082       73   FORMAT('0ALPHA= ',G12.4)
```

```
0003                C
0004                C     INTIALIZE S,BETA(1),T(1),IFRST,IBLK1,NREM,AND NREAD
0005                C
0006                      DO 200 ITME1=IBUF1+1,IBUF2
0007                        S(ITME1)=0
0008          200   CONTINUE
0009                C
0010                      BETA(1)=0.
0011                      T(1)=1
0012                C
0013                C     IFRST POINTS TO THE FIRST SAMPLE IN THE BLOCK.
0014                C
0015                      IFRST=1
0016                C
0017                C     IBLK1 IS THE BLOCK NUMBER.
0018                C
0019                      IBLK1=1
0020                C
0021                C     NREM IS THE NUMBER OF SAMPLES REMAINING IN THE BUFFER.
0022                C
0023                      NREM=0
0024                C
0025                C     NREAD IS THE NUMBER OF "COMPLETE BUFFER" READS.
0026                C
0027                      NREAD=NSAMP/IBUF1
0028                      WRITE(6,74)
0029           74   FORMAT('0BLOCK',5X,'BETA',6X,'T')
```

```
FORTRAN IV-PLUS V02-51          15:47:32     15-APR-80          PAGE 6
REPEAT.FTN      /TR:BLOCKS/WR


0094            IF(NREAD.LE.0)GO TO 210
0095                    DO 220 IREAD=1,NREAD
        C
        C                       IF(IREAD IS EVEN)GO TO 221
        C
0096                            IF(IAND(IREAD,1).EQ.0)GO TO 221
0097                                    READ(1,230)(S(ITME2),ITME2=1,IBUF1)
0098    230                             FORMAT(16I5)
        C
        C                               DO WHILE(MODB2(IFRST+KBLK-1).LE.IBUF1)
        C
0099    250                             IF(MODB2(IFRST+KBLK-1).GT.IBUF1)
               1                        GO TO 240
0100                                            CALL PITCH(BETA(IBLK1),T(IBLK1),
               1                                MODB2(IFRST))
0101                                            WRITE(6,600)IBLK1,BETA(IBLK1),
               1                                T(IBLK1)
0102    600                                     FORMAT(2X,I3,2X,G11.4,I4)
0103                                            IBLK1=IBLK1+1
0104                                            IFRST=MODB2(IFRST+KBLK)
0105                                    GO TO 250
0106    240                             CONTINUE
0107                            GO TO 260
```

```
0108    221                         READ(1,230)(S(ITME3),ITME3=IBUF1+1,
                1                   IBUF2)
        C
        C                           DO WHILE(IFRST+KBLK-1.LE.IBUF2)
        C
0109    280                         IF(IFRST+KBLK-1.GT.IBUF2)GO TO 270
0110                                        CALL PITCH(BETA(IBLK1),T(IBLK1),
                1                           IFRST)
0111                                        WRITE(6,600)IBLK1,BETA(IBLK1),
                1                           T(IBLK1)
0112                                        IBLK1=IBLK1+1
0113                                        IFRST=IFRST+KBLK
0114                                GO TO 280
0115    270                         CONTINUE
0116                        GO TO 260
0117    260                 CONTINUE
0118    220             CONTINUE
        C               NREM=IBUF1-MODB1(IFRST)+1                       I-I.10
0119                    NREM=IAND(IBUF1-MODB1(IFRST)+1,IBUF1-1)         II.10
0120            GO TO 210
0121    210     CONTINUE
```

```
FORTRAN IV-PLUS V02-51          15:47:32    15-APR-80           PAGE 8
REPEAT.FTN      /TR:BLOCKS/WR


0122            NREM1=IAND(NSAMP,IBUF1-1)
      C
      C       IF(TOTAL SAMPLES REMAINING.LT.KBLK)GO TO  290
      C
0123            IF(NREM1+NREM.LT.KBLK)GO TO 290
      C
      C               IF((NREAD+1)IS EVEN) GO TO 300
      C
0124                    IF(IAND(NREAD,1).EQ.1)GO TO 300
0125                            READ(1,230)(S(ITME4),ITME4=1,NREM1)
      C
      C                         DO WHILE(MODB2(IFRST+KBLK-1).LE.IBUF1)
      C
0126    320                     IF(MODB2(IFRST+KBLK-1).GT.IBUF1)GO TO 310
0127                                    CALL PITCH(BETA(IBLK1),T(IBLK1),
              1                         MODB2(IFRST))
0128                                    WRITE(6,600)IBLK1,BETA(IBLK1),
              1                         T(IBLK1)
0129                                    IBLK1=IBLK1+1
0130                                    IFRST=MODB2(IFRST+KBLK)
0131                            GO TO 320
0132    310                     CONTINUE
0133                    GO TO 330
```

```
                                READ(1,230)(S(ITME5),ITME5=IBUF1+1,IBUF1+NREM1)
                C
                C               DO WHILE(IFRST+KBLK-1.LE.IBUF2)
                C
                                IF(IFRST+KBLK-1.GT.IBUF2)GO TO 340
                                        CALL PITCH(BETA(IBLK1),T(IBLK1),IFRST)
                                        WRITE(6,600)IBLK1,BETA(IBLK1),T(IBLK1)
                                        IBLK1=IBLK1+1
                                        IFRST=IFRST+KBLK
                                GO TO 350
                340             CONTINUE
                        GO TO 330
                330     CONTINUE
                GO TO 290
                290     CONTINUE
                C
                C       IF MODB1(IFRST)=1, THERE ARE NO LEFTOVER SAMPLES.
                C
                        IF(MODB1(IFRST).EQ.1)GO TO 360
                C
                C               SET BETA=0.,T=1, FOR LEFTOVER SAMPLES.
                C
                                BETA(IBLK1)=0.
                                T(IBLK1)=1
                                WRITE(6,600)IBLK1,BETA(IBLK1),T(IBLK1)
                GO TO 360
                360     CONTINUE
```

```
FORTRAN IV-PLUS V02-51          15:47:32     15-APR-80          PAGE 10
REPEAT.FTN      /TR:BLOCKS/WR


        C
        C       CLOSE THE SPEECH FILE
        C
0152            CLOSE(UNIT=1)
        C
        C       OPEN THE POINTER FILE FOR PARAMETERS.
        C
0153            OPEN(UNIT=3,NAME=FNAME1,TYPE='OLD',READONLY,SHARED)
0154            READ(3,370)NRUNS
0155    370     FORMAT(I2)
        C
        C  75   TYPE *,' IF YOU WISH TO RUN AGAIN,TYPE 1'
        C       ACCEPT *,IRUN
        C       IF(IRUN.NE.1)GOTO 110
0156            DO 76 IRUNS=1,NRUNS
        C       CALL SUBROUTINE INSTRT WHICH RETURNS VALUE OF L.
        C
0157            CALL INSTRT(L)
        C
        C       REOPEN THE SPEECH FILE AND SKIP OVER THE HEADER.
        C
0158            OPEN(UNIT=1,NAME=FNAME,TYPE='OLD',READONLY,SHARED)
0159            READ(1,400)
0160    400     FORMAT(80X)
        C
        C       OPEN THE OUTPUT (SHAT) FILE AND WRITE THE HEADER
        C
0161            OPEN(UNIT=2,NAME=FNAME2,TYPE='NEW',CARRIAGECONTROL='LIST')
0162            WRITE(2,401)NSENT,IRATE,NSAMP,IUPPR,ILOWR,NTERMS,ICOMM
0163    401     FORMAT(6I5,10X,40A1)
0164            IF(QPLOT.NE.'Y')GO TO 407
0165                    OPEN(UNIT=4,NAME=FNAME3,TYPE='NEW',
                1        CARRIAGECONTROL='LIST')
0166                    WRITE(4,401)NSENT,IRATE,NSAMP,IUPPR,ILOWR,NTERMS,ICOMM
0167                    ICNT=0
0168    407     CONTINUE
```

```
                C
                C       INITIALIZE ES,EN,SHAT,VHAT,S
                C
0169                    ES=0.
0170                    EN=0.
0171                    DO 410 ITM1=1,IBUF1
0172                            SHAT(ITM1)=0.
0173                            VHAT(ITM1)=0.
0174                            S(IBUF1+ITM1)=0
0175            410     CONTINUE
0176                    WRITE(6,620)
0177            620     FORMAT('0SAMPLE',5X,'ES',10X,'EN')
0178                    BUFCNT=0.                                       11.2
0179                    PREFIX=0                                        11.2
```

```
FORTRAN IV-PLUS V02-51          15:47:32     15-APR-80          PAGE 12
REPEAT.FTN      /TR:BLOCKS/WR


0180            NREAD2=NSAMP/IBUF1
0181            IF(NREAD2.LE.0)GO TO 420
0182                    DO 430 IREAD2=1,NREAD2
        C
        C                       IF(IREAD2 IS EVEN)GO TO 431
        C
0183                            IF(IAND(IREAD2,1).EQ.0)GO TO 431
0184                                    READ(1,440)(S(ITM2),ITM2=1,IBUF1)
0185    440                             FORMAT(1615)
0186                                    DO 450 ITM3=1,IBUF1
0187                                            ITM4=(IREAD2-1)*IBUF1+ITM3
0188                                            CALL LPPROC(ALPHA,BETA,BUFCNT,  !1.2
              1                              BUFFIL,EN,ES,IBUF1,ITM3,ITM4,   !1.2
              2                              ITM5,ITM7,KBLK,L,LUNBUF,LUNPRT, !1.2
              3                              LUNSH,MXBLCK,PREFIX,SHAT,T,V,   !1.2
              4                              VHAT)                           !1.2
        C                                   CALL LPPROC(ALPHA,BETA,EN,ES,   !-1.2
        C     1                              IBUF1,ITM3,ITM4,ITM5,KBLK,L,    !-1.2
        C     2                              LUNPRT,LUNSH,MXBLCK,SHAT,T,V,   !-1.2
        C     3                              VHAT)                           !-1.2
        C                                   ITM5=ITM4-L+1                   !-1.1
        C                                   ITM6=MODB1(ITM4)                !-1.1
        C                                   ITM7=MODB1(ITM5)                !-1.1
        C                                   IBLK2=(ITM4-1)/KBLK+1           !-1.1
        C                                   IBLK3=(ITM5-1)/KBLK+1           !-1.1
        C                                                                   !-1.1
        C                                   CONSTRUCT THE PITCH-REDUCED     !-1.1
        C                                   SPEECH.                         !-1.1
        C                                                                   !-1.1
        C                                   V(ITM6)=FLOAT(S(ITM3))-ALPHA    !-1.1
        C     1                             *BETA(IBLK2)*SHAT(MODB1(ITM3    !-1.1
        C     2                             -T(IBLK2)))                     !-1.1
        C                                                                   !-1.1
        C                                   CALL INLOOP TO GET Q, VHAT      !-1.1
        C                                                                   !-1.1
        C                                   CALL INLOOP(V(ITM6),Q,VHAT      !-1.1
        C     1                             (ITM7))                         !-1.1
```

```
C                                                      1-1.1
C                          RECONSTRUCT THE SPEECH.     1-1.1
C                                                      1-1.1
C                          IF(ITM5.LE.0)GO TO 460      1-1.1
C                                  BTA=BETA(IBLK3)     1-1.1
C                                  IT1=T(IBLK3)        1-1.1
C                          GO TO 470                   1-1.1
C 460                              BTA=0.              1-1.1
C                                  IT1=1               1-1.1
C                          GO TO 470                   1-1.1
C 470                      CONTINUE                    1-1.1
C                          SHAT(ITM7)=VHAT(ITM7)+ALPHA 1-1.1
C        1                 *BTA*SHAT(MODB1(ITM7-IT1))  1-1.1
C                          SHAT(ITM7)=AMAX1(SHAT(ITM7), 1-1.1
C        1                 -2048.)                     1-1.1
C                          SHAT(ITM7)=AMIN1(SHAT(ITM7), 1-1.1
C        1                 2047.)                      1-1.1
C                          IF((ITM7.NE.IBUF1).OR.(ITM5 1-1.1
C        1                 .LE.0))GO TO 473            1-1.1
C                                  WRITE(2,474)(NINT(  1-1.1
C        1                         SHAT(IXI)),IXI=1,IBUF1) 1-1.1
C 474                              FORMAT(16I5)        1-1.1
C                          GO TO 473                   1-1.1
C 473                      CONTINUE                    1-1.1
```

```
FORTRAN IV-PLUS V02-51          15:47:32     15-APR-80          PAGE 14
REPEAT.FTN      /TR:BLOCKS/WR

C                                                                      I-1.1
C                                                                      I-1.1
C                                       CALCULATE THE SIGNAL AND NOISE I-1.1
C                                       ENERGIES.                      I-1.1
C                                                                      I-1.1
C                                       FS1=FLOAT(S(MODB2(ITM5)))      I-1.1
C                                       ES=ES+FS1**2                   I-1.1
C                                       EN=EN+(FS1-SHAT(ITM7))**2      I-1.1
C                                       IF(MOD(ITM5,100).NE.0)GO TO 630I-1.1
C 610                                           WRITE(6,610)ITM5,ES,EN I-1.1
C                                               FORMAT(I7,2G12.4)      I-1.1
C 630                                   GO TO 630                      I-1.1
#189    450                             CONTINUE                       I-1.1
#190                            GO TO 432
```

```
#191     431                          READ(1,440)(S(ITM2),ITM2=IBUF1+1,IBUF2)
#192                                   DO 471 ITM3=IBUF1+1,IBUF2
#193                                           ITM4=(IREAD2-2)*IBUF1+ITM3
#194                                           CALL LPPROC(ALPHA,BETA,BUFCNT,   !1.2
                 1                              BUFFIL,EN,ES,IBUF1,ITM3,ITM4,   !1.2
                 2                              ITM5,ITM7,KBLK,L,LUNBUF,LUNPRT,!1.2
                 3                              LUNSH,MXBLCK,PREFIX,SHAT,T,V,   !1.2
                 4                              VHAT)                           !1.2
          C                                    CALL LPPROC(ALPHA,BETA,EN,ES,    !-1.2
          C      1                              IBUF1,ITM3,ITM4,ITM5,KBLK,L,    !-1.2
          C      2                              LUNPRT,LUNSH,MXBLCK,SHAT,T,V,   !-1.2
          C      3                              VHAT)                           !-1.2
          C                                    ITM5=ITM4-L+1                    !-1.1
          C                                    ITM6=MOD81(ITM4)                 !-1.1
          C                                    ITM7=MOD81(ITM5)                 !-1.1
          C                                    IBLK2=(ITM4-1)/KBLK+1            !-1.1
          C                                    IBLK3=(ITM5-1)/KBLK+1            !-1.1
          C                                    V(ITM6)=FLOAT(S(ITM3))-ALPHA     !-1.1
          C      1                             *BETA(IBLK2)*SHAT(MOD81(ITM3     !-1.1
          C      2                             -T(IBLK2)))                      !-1.1
          C                                    CALL INLOOP(V(ITM6),Q            !-1.1
          C      1                             ,VHAT(ITM7))                     !-1.1
          C                                    SHAT(ITM7)=VHAT(ITM7)+ALPHA      !-1.1
          C      1                             *BETA(IBLK3)*SHAT(MOD81(ITM7     !-1.1
          C      2                             -T(IBLK3)))                      !-1.1
          C                                    SHAT(ITM7)=AMAX1(SHAT(ITM7),     !-1.1
          C      1                             -2048.)                          !-1.1
          C                                    SHAT(ITM7)=AMIN1(SHAT(ITM7),     !-1.1
          C      1                             2047.)                           !-1.1
          C                                    IF(ITM7.EQ.IBUF1)WRITE(2,474)    !-1.1
          C      1                             (NINT(SHAT(IX2)),IX2=1,IBUF1)    !-1.1
          C                                    FS1=FLOAT(S(MOD82(ITM5)))        !-1.1
          C                                    ES=ES+FS1**2                     !-1.1
          C                                    EN=EN+(FS1-SHAT(ITM7))**2        !-1.1
          C                                    IF(MOD(ITM5,1000).EQ.0)WRITE(6,  !-1.1
          C      1                             610)ITM5,ES,EN                   !-1.1
#195     471                          CONTINUE
#196                            GO TO 432
#197     432                    CONTINUE
#198     430               CONTINUE
#199               GO TO 420
#200     420       CONTINUE
```

```
FORTRAN IV-PLUS V02-51          15:47:32    15-APR-80          PAGE 16
REPEAT.FTN      /TR:BLOCKS/WR


0201            NREAD3=IAND(NSAMP,IBUF1-1)
0202            IF(NREAD3.EQ.0)GO TO 481                                    11.10
      C
      C         IF(NREAD2+1 IS EVEN)GO TO 472
      C
0203            IF(IAND(NREAD2,1).EQ.1)GO TO 472
0204                    READ(1,440)(S(ITM2),ITM2=1,NREAD3)
0205                    DO 480 ITM3=1,NREAD3
0206                            ITM4=NREAD2*IBUF1+ITM3
0207                            CALL LPPROC(ALPHA,BETA,BUFCNT,BUFFIL,EN,ES,     11.2
               1                 IBUF1,ITM3,ITM4,ITM5,ITM7,KBLK,L,LUNBUF,LUNPRT,11.2
               2                 LUNSH,MXBLCK,PREFIX,SHAT,T,V,VHAT)            11.2
      C                         CALL LPPROC(ALPHA,BETA,EN,ES,IBUF1,ITM3,ITM4,   1-1.2
      C         1                 ITM5,KBLK,L,LUNPRT,LUNSH,MXBLCK,SHAT,T,V,VHAT) 1-1.2
      C                         ITM5=ITM4-L+1                                   1-1.1
      C                         ITM6=MODB1(ITM4)                                1-1.1
      C                         ITM7=MODB1(ITM5)                                1-1.1
      C                         IBLK2=(ITM4-1)/KBLK+1                           1-1.1
      C                         IBLK3=(ITM5-1)/KBLK+1                           1-1.1
      C                         V(ITM6)=FLOAT(S(ITM3))-ALPHA*BETA(IBLK2)*SHAT   1-1.1
      C         1               (MODB1(ITM3-T(IBLK2)))                          1-1.1
      C                         CALL INLOOP(V(ITM6),Q,VHAT(ITM7))               1-1.1
      C                         IF(ITM5.LE.0)GO TO 490                          1-1.1
      C                                 BTA=BETA(IBLK3)                         1-1.1
      C                                 IT1=T(IBLK3)                            1-1.1
      C                         GO TO 500                                       1-1.1
      C 490                             BTA=0.                                  1-1.1
      C                                 IT1=1                                   1-1.1
      C                         GO TO 500                                       1-1.1
      C 500                     CONTINUE                                        1-1.1
      C                         SHAT(ITM7)=VHAT(ITM7)+ALPHA*BTA*SHAT(MODB1(ITM7 1-1.1
      C         1               -IT1))                                          1-1.1
      C                         SHAT(ITM7)=AMAX1(SHAT(ITM7),-2048.)             1-1.1
      C                         SHAT(ITM7)=AMIN1(SHAT(ITM7),2047.)              1-1.1
```

```
C                       IF((ITM7.EQ.IBUF1).AND.(ITM5.GT.0))WRITE(2,474)  I-1.1
C       1               (NINT(SHAT(IX3)),IX3=1,IBUF1)                    I-1.1
C                       FS1=FLOAT(S(MODB2(ITM5)))                        I-1.1
C                       ES=ES+FS1**2                                     I-1.1
C                       EN=EN+(FS1-SHAT(ITM7))**2                        I-1.1
C                       IF(MOD(ITM5,1000).EQ.0)WRITE(6,610)ITM5,ES,EN    I-1.1
0208    480             CONTINUE
0209            GO TO 481
0210    472             READ(1,440)(S(ITM2),ITM2=IBUF1+1,IBUF1+NREAD3)
0211                    DO 482 ITM3=IBUF1+1,IBUF1+NREAD3
0212                    ITM4=(NREAD2-1)*IBUF1+ITM3
0213                    CALL LPPROC(ALPHA,BETA,BUFCNT,BUFFIL,EN,ES,      I1.2
        1                 IBUF1,ITM3,ITM4,ITM5,ITM7,KBLK,L,LUNBUF,LUNPRT,I1.2
        2                 LUNSH,MXBLCK,PREFIX,SHAT,T,V,VHAT)             I1.2
C                       CALL LPPROC(ALPHA,BETA,EN,ES,IBUF1,ITM3,ITM4,    I-1.2
C       1                 ITM5,KBLK,L,LUNPRT,LUNSH,MXBLCK,SHAT,T,V,VHAT) I-1.2
C                       ITM5=ITM4-L+1                                    I-1.1
C                       ITM6=MODB1(ITM4)                                 I-1.1
C                       ITM7=MODB1(ITM5)                                 I-1.1
C                       IBLK2=(ITM4-1)/KBLK+1                            I-1.1
C                       IBLK3=(ITM5-1)/KBLK+1                            I-1.1
C                       V(ITM6)=FLOAT(S(ITM3))-ALPHA*BETA(IBLK2)         I-1.1
C       1               *SHAT(MODB1(ITM3-T(IBLK2)))                      I-1.1
C                       CALL INLOOP(V(ITM6),Q,VHAT(ITM7))                I-1.1
C                       SHAT(ITM7)=VHAT(ITM7)+ALPHA*BETA(IBLK3)          I-1.1
C       1               *SHAT(MODB1(ITM7-T(IBLK3)))                      I-1.1
C                       SHAT(ITM7)=AMAX1(SHAT(ITM7),-2048.)              I-1.1
C                       SHAT(ITM7)=AMIN1(SHAT(ITM7),2047.)               I-1.1
C                       IF(ITM7.EQ.IBUF1)WRITE(2,474)(NINT(SHAT(IX4)),   I-1.1
C       1               IX4=1,IBUF1)                                     I-1.1
C                       FS1=FLOAT(S(MODB2(ITM5)))                        I-1.1
C                       ES=ES+FS1**2                                     I-1.1
C                       EN=EN+(FS1-SHAT(ITM7))**2                        I-1.1
C                       IF(MOD(ITM5,1000).EQ.0)WRITE(6,610)ITM5,ES,EN    I-1.1
0214    482             CONTINUE
0215            GO TO 481
0216    481     CONTINUE
```

```
FORTRAN IV-PLUS V02-51          15:47:32    15-APR-80          PAGE 18
REPEAT.FTN      /TR:BLOCKS/WR


        C
        C       CLOSE THE SPEECH FILE.
        C
#217            CLOSE(UNIT=1)
        C
        C       FINISH AND CLOSE THE OUTPUT FILE
        C
#218            IF(L.EQ.1)GO TO 484
#219                    DO 485 IX5=1,L-1
#220                            ITM8=MODB1(ITM5+IX5)
#221                            SHAT(ITM8)=0.
#222                            IF(ITM8.EQ.IBUF1)WRITE(2,474)(NINT(SHAT(IX6)),
                1               IX6=1,IBUF1)
#223    474                     FORMAT(16I5)
#224    485             CONTINUE
#225            GO TO 486
#226    484             ITM8=ITM7
#227            GO TO 486
#228    486     CONTINUE
#229            IF(ITM8.NE.IBUF1)WRITE(2,474)(NINT(SHAT(IX7)),IX7=1,ITM8)
#230            CLOSE(UNIT=2)
#231            IF(.NOT.((QPLOT.EQ.'Y').AND.(ICNT.NE.0)))GO TO 488
#232                    WRITE(4,487)(MBUFF(IX8),IX8=1,ICNT)
#233    487             FORMAT(<ICNT>I5)
#234    488     CONTINUE
#235            IF(QPLOT.NE.'Y')GO TO 489
#236                    CLOSE(UNIT=4)
#237    489     CONTINUE
#238            IF(BUFFIL.EQV.NO)GO TO 490                                      11.2
#239                    CLOSE(UNIT=LUNBUF)                                      11.2
#240            GO TO 490                                                       11.2
#241    490     CONTINUE                                                        11.2
        C
        C       COMPUTE SIGNAL TO NOISE RATIO.
        C
#242            WRITE(6,610)NSAMP,ES,EN
#243    610     FORMAT(I7,2G12.4)
#244            SNR=10.*ALOG10(ES/EN)
        C
        C
        C
#245            WRITE(6,90) SNR
#246    90      FORMAT('0SNR OVERALL= ',G11.4,' DB')
#247            WRITE(5,90)SNR
#248            CALL INEND
```

```
        C       GOTO 75
        C 110   STOP
#249    76      CONTINUE
        C
        C       CLOSE THE POINTER FILE FOR PARAMETERS.
        C
#250            CLOSE(UNIT=3)
#251            CALL DATE(IDATE)
#252            CALL TIME(ITIME)
#253            WRITE(6,11)IDATE,ITIME
#254            STOP
#255            END
```

```
FORTRAN IV-PLUS V02-51          15:32:32    15-APR-80          PAGE 1
PITCH.FTN       /TR:BLOCKS/WR

0001            SUBROUTINE PITCH(BETA,T,IJ)
                C
                C       PURPOSE:  TO CALCULATE THE PITCH PERIOD T AND THE PREDICTION
                C                  COEFFICIENT BETA.
                C
                C       V.R     DATE            NAME            COMMENTS
                C       0.1     16-JUL-79       J.M. KRESSE     EXISTING SOFTWARE
                C       1.0     17-JUL-79       J.M. KRESSE     BACKED UP
                C
                C
                C       THIS IS A PITCH EXTRACTION PROGRAM.
                C       BASIC PURPOSE OF PITCH EXTRACTION LOOP IN OUR ALGORITHM IS
                C       TO REMOVE PITCH REDUNDANCY OF SPEECH SIGNAL.HERE THE
                C       BLOCK ADAPTATION IS ASSUMED WITH BLOCK LENGTH K.
                C       K VARIES BETWEEN
                C       80 AND 200.
                C
                C       PARAMETER IBUF2=512,ITL=20,ITU=150,KBLK=150                1-1.7
0002            PARAMETER IBUF2=512,ITL=20,ITU=150,KBLK=100                11.7
0003            INTEGER S,T
0004            COMMON S(IBUF2)
0005            DIMENSION A(ITU),B(ITU)
```

```
0006            MODB2(INDX1)=IAND(INDX1-1,IBUF2-1)+1
        C
        C       CALCULATE CORRELATION BETWEEN SAMPLES.
        C
0007            BIG=0.
0008            IK=IJ+KBLK-1
0009            DO 60 IT=ITL,ITU
0010            SUM1=0.
0011            SUM2=0.
0012            DO 50 J=IJ,IK
0013            M=S(MODB2(J))
0014            MM=S(MODB2(J-IT))
0015            F=FLOAT(M)
0016            FF=FLOAT(MM)
0017            SUM1=SUM1+F*FF
0018            SUM2=SUM2+FF*FF
0019         50 CONTINUE
0020            IF(SUM2.LT..0001)GOTO 51
0021            B(IT)=SUM1/SUM2
0022            A(IT)=B(IT)*SUM1
0023            IF(BIG.LT.A(IT))BIG=A(IT)
0024            GOTO 60
0025         51 B(IT)=0.
0026            A(IT)=0.
0027         60 CONTINUE
```

```
[illegible] IV-PLUS V02-51          15:32:32    15-APR-80          PAGE 3
[illegible]      /TR:BLOCKS/WR

                C
                C       FIND THE LARGEST ELEMENT OF ARRAY.
                C
0028                    DO 70 I=ITL,ITU
0029                    I1=I
0030                    IF(BIG.EQ.A(I)) GOTO 80
0031            70      CONTINUE
0032            80      T=I1
0033                    B(I1)=AMAX1(B(I1),-2.0)
0034                    B(I1)=AMIN1(B(I1),2.0)
0035                    BETA=B(I1)
0036                    RETURN
0037                    END
```

```
0001            SUBROUTINE INSTRT(L1)
                C
                C       PURPOSE:  TO INITIALIZE THE INLOOP SYSTEM.
                C
                C       V.R     DATE            NAME            COMMENTS
                C       0.1     16-JUL-79       J.M. KRESSE     EXISTING SOFTWARE
                C       1.0     17-JUL-79       J.M. KRESSE     BACKED UP
                C       1.3A    19-SEP-79       J.M. KRESSE     ADD BUFFER CONTROL
                C       1.4     26-SEP-79       J.M. KRESSE     ADD BUFFER OCCUPANCY
                C
                C       SUBROUTINE INSTRT IS USED TO READ IN PARAMETER VALUES AND
                C       INITIALIZE QUANTITIES FOR SUBROUTINE INLOOP.
                C
0002            PARAMETER MXQM=11,MXN=8,MXM=1,MXL=10,MXLN=10
0003            PARAMETER HINOIS=.FALSE.,LONOIS=.TRUE.
0004            PARAMETER NRANGE=10                                     11.3A
0005            BYTE FNAME(33),IDATE(10),ITIME(9)
0006            INTEGER BUFOCC                                          11.4
0007            INTEGER QLVL,QMAX,QN,QO,RANK
0008            LOGICAL LEVEL
0009            LOGICAL GAPPED                                          11.6
0010            REAL LOSNR,NOISE
0011            REAL BFFCAR,RANGE,T                                     11.3A
0012            COMMON /ALPHA/ ALPHA(MXQM)
0013            COMMON /AN/ AN(MXQM*MXM,MXN)
0014            COMMON /AO/ AO(MXM,MXN)
0015            COMMON /B/ B(MXN)
0016            COMMON /BFFCAR/ BFFCAR(NRANGE)                          11.3A
0017            COMMON /BUFOCC/ BUFOCC(NRANGE)                          11.4
0018            COMMON /CN/ CN(MXQM*MXM)
0019            COMMON /EAASN/ EAASN(MXQM*MXM)
0020            COMMON /EAASO/ EAASO(MXM)
0021            COMMON /F/ F(MXQM)
0022            COMMON /GAPPED/ GAPPED                                  11.6
0023            COMMON /INBUF/ VN(MXL)
0024            COMMON /LEFDIS/ LEFDIS(MXL)
0025            COMMON /MADAPT/ FACTOR,HISNR,LOSNR
0026            COMMON /MEFDIS/ MEFDIS(MXM)
0027            COMMON /MVARY/ LEVEL,M1,NLONO,NHINO
0028            COMMON /NNODEN/ NNODEN
0029            COMMON /NNODEO/ NNODEO
0030            COMMON /PARAM/ M,L,N,QMAX,MXVHN,DELTA,ALPH,EAASND,EAASNM,G
0031            COMMON /QLVL/ QLVL(MXQM)
0032            COMMON /QN/ QN(MXQM*MXM,MXL)
0033            COMMON /QO/ QO(MXM,MXL-1)
0034            COMMON /RANGE/ RANGE(NRANGE+1)                          11.3A
0035            COMMON /RANK1/ RANK(MXQM*MXM)
0036            COMMON /SIGMN/ SIGMN(MXQM*MXM)
0037            COMMON /SIGMO/ SIGMO(MXM)
0038            COMMON /STATS/ NSAMP,VSQR,NOISE
0039            COMMON /T/ T(MXQM,2)                                    11.3A
0040            COMMON /VHATN/ VHATN(MXQM*MXM,MXLN)
0041            COMMON /VHATO/ VHATO(MXM,MXLN-1)
                C
                C
```

```
FORTRAN IV-PLUS V02-51          16:55:14     02-MAY-80          PAGE 2
INSTR5.FTN      /TR:BLOCKS/WR

        C
        C       PRINT DATE AND TIME
        C
0042            CALL DATE(IDATE)
0043            CALL TIME(ITIME)
0044            WRITE(6,10)IDATE,ITIME
0045    10      FORMAT('0DATE: ',10A1,' TIME: ',9A1)
```

```
C
C       READ IN FILENAME OF FILE CONTAINING PARAMETERS, INITIALIZATION
C       DATA.
C
C       TYPE *,' ENTER FILENAME OF PARAMETER FILE: '
C       ACCEPT 10,NLETTR,(FNAME(ILETTR),ILETTR=1,NLETTR)
0046            READ(3,20)FNAME
0047    20      FORMAT(33A1)
C
C       INSERT END-OF-STRING MARK.
C
0048            FNAME(33)=0
0049            WRITE(6,30)FNAME
0050    30      FORMAT('0PARAMETER FILENAME: ',33A1)
0051            OPEN (UNIT=8,NAME=FNAME,TYPE='OLD',READONLY)
C
C       READ IN SCALAR PARAMETERS
C
0052            READ(8,40)L,M,N,QMAX,DELTA,ALPH,EAASND,EAASNM,G
0053    40      FORMAT(4I3,5G15.7)
0054            MXVHN=MAX0(L-1,N)
0055            WRITE(6,50)L,M,N,QMAX,MXVHN,DELTA,ALPH,EAASND,EAASNM,G
0056    50      FORMAT('0SCALAR PARAMETERS:'/' L=',I3,' M=',I3,' N=',I3,
                1' QMAX=',I3,' MXVHN=',I3/' DELTA=',G12.4,' ALPH=',G12.4,
                2' EAASND=',G12.4,' EAASNM=',G12.4,' G=',G12.4)
0057            L1=L
```

```
C
C       READ IN ARRAY PARAMETERS
C
0058            READ (8,60) (F(IQ1),IQ1=1,QMAX),(ALPHA(IQ2),IQ2=1,QMAX),
                1(B(ICOEF),ICOEF=1,N)
0059    60      FORMAT(<QMAX>F7.2/<QMAX>F7.2/<N>F7.2)
0060            WRITE(6,70)(F(IQ3),IQ3=1,QMAX),(ALPHA(IQ4),IQ4=1,QMAX),
                1(B(ICOEF1),ICOEF1=1,N)
0061    70      FORMAT('0ARRAY PARAMETERS:'/' F:',4X,<QMAX>F7.2/' ALPHA:',
                1<QMAX>F7.2/' B=',4X,<N>F7.2)
C
C       READ IN INITIAL VALUES FOR SIGMA, COEFFICIENTS
C
0062            READ (8,80) SIGMO(1),(AO(1,ICOEF2),ICOEF2=1,N)
0063    80      FORMAT(F7.2/<N>F7.2)
0064            WRITE(6,90)SIGMO(1),(AO(1,ICOEF3),ICOEF3=1,N)
0065    90      FORMAT('0INITIAL CONDITIONS:'/' SIGMA=',4X,F7.2/' PREDICTOR:',
                1<N>F7.2)
C
C       READ IN ADAPTIVE M PARAMETERS
C
0066            READ(8,*)FACTOR,HISNR,LOSNR
0067            WRITE(6,95)FACTOR,HISNR,LOSNR
0068    95      FORMAT('0ADAPTIVE M PARAMETERS:'/
                1 ' FACTOR=',G12.4,'; HISNR=',G12.4,'; LOSNR=',G12.4)
0069            READ(8,*)((T(IX3,IX4),IX4=1,2),IX3=1,QMAX)                    11.3A
0070            READ(8,*)RANGE                                                11.3A
0071            READ(8,*)BFFCAR                                               11.3A
0072            WRITE(6,*)' T:',((T(IX3,IX4),IX4=1,2),IX3=1,QMAX)             11.3A
0073            WRITE(6,*)' RANGE:',RANGE                                     11.3A
0074            WRITE(6,*)' BFFCAR:',BFFCAR                                   11.3A
C
C       CLOSE FILE
C
0075            CLOSE (UNIT=8)
```

```
C
C       COMPLETE INITIALIZATION OF THE TREE.
C
0076            NNODEO=1
0077            EAASO(1)=0.
0078            DO 100 IBCK1=1,MXVHN
0079                    VHATO(1,IBCK1)=0.
0080    100     CONTINUE
0081            IF(L.LE.1)GO TO 120
0082                    DO 110 IBCK=1,L-1
0083                            QO(1,IBCK)=1
0084                            VN(IBCK)=0.
0085    110             CONTINUE
0086    120     CONTINUE
C
C       INITIALIZE THE STATISTICAL COUNTERS.
C
0087            NSAMP=0
0088            VSQR=0.
0089            NOISE=0.
0090            DO 130 IQ3=1,QMAX
0091                    QLVL(IQ3)=0
0092    130     CONTINUE
0093            LEVEL=HINOIS
0094            NLONO=0
0095            NHINO=0
0096            M1=M
0097            DO 140 IX1=1,L
0098                    LEFDIS(IX1)=0
0099    140     CONTINUE
0100            DO 150 IX2=1,M
0101                    MEFDIS(IX2)=0
0102    150     CONTINUE
0103            DO 160 IX5=1,NRANGE
0104                    BUFOCC(IX5)=0
0105    160     CONTINUE
0106            GAPPED=.FALSE.
0107            RETURN
0108            END
```

```
FORTRAN IV-PLUS V02-51          15:34:16    15-APR-80          PAGE 1
LPPROC.FTN      /TR:BLOCKS/WR

0001            SUBROUTINE LPPROC(ALPHA,BETA,BUFCNT,BUFFIL,EN,ES,IBUF1,ITM3,     11.2
                1 ITM4,ITM5,ITM7,KBLK,L,LUNBUF,LUNPRT,LUNSH,MXBLCK,PREFIX,       11.5A
                2 SHAT,T,V,VHAT)                                                 11.5A
      C         SUBROUTINE LPPROC(ALPHA,BETA,EN,ES,IBUF1,ITM3,ITM4,ITM5,KBLK,    1-1.2
      C         1 L,LUNPRT,LUNSH,MXBLCK,SHAT,T,V,VHAT)                           1-1.2
      C
      C         PURPOSE:  TO PERFORM THE LOOP PROCESSING.
      C
      C         V.R     DATE            NAME            COMMENTS
      C         1.1     19-JUL-79       J.M. KRESSE     ORIGINATION
      C         1.2     22-JUL-79       J.M. KRESSE     OUTPUT BUFFER COUNTER
      C         1.3A    19-SEP-79       J.M. KRESSE     ADD BUFFER CONTROL
      C
      C--------DECLARATIONS
      C
0002            PARAMETER IBUF2=512,MAXVAL=2047.,MINVAL=-2048.
0003            PARAMETER NO=.FALSE.,YES=.TRUE.                                  11.2
0004            PARAMETER BUFTHR=800.,MXQM=11                                    11.6
0005            INTEGER IBLK2,IBLK3,IBUF1,ITM3,ITM4,ITM5,ITM6,ITM7,IT1,IX1,KBLK
0006            INTEGER L,LUNBUF,LUNPRT,LUNSH,MXBLCK,PREFIX,Q,S,T(MXBLCK)        11.2
      C         INTEGER L,LUNPRT,LUNSH,MXBLCK,Q,S,T(MXBLCK)                      1-1.2
0007            LOGICAL BUFFIL                                                   11.2
0008            LOGICAL GAPPED                                                   11.6
0009            REAL ALPHA,BETA(MXBLCK),BTA,BUFCNT,EN,ES,FS1,SHAT(IBUF1)         11.2
0010            REAL V(IBUF1),VHAT(IBUF1)                                        11.2
      C         REAL ALPHA,BETA(MXBLCK),BTA,EN,ES,FS1,SHAT(IBUF1),V(IBUF1)       1-1.2
      C         REAL VHAT(IBUF1)                                                 1-1.2
0011            REAL BTASAV,THR,THRSAV(MXQM,2)                                   11.6
0012            COMMON S(IBUF2)
0013            COMMON /GAPPED/ GAPPED                                           11.6
0014            COMMON /T/ THR(MXQM,2)                                           11.6
      C
      C--------DEFINITIONS
      C
0015            ITM5=ITM4-L+1
0016            ITM6=MODB(ITM4,IBUF1)
0017            ITM7=MODB(ITM5,IBUF1)
0018            IBLK2=((ITM4-1)/KBLK)+1
0019            IBLK3=((ITM5-1)/KBLK)+1
      C
      C--------PROCEDURE
      C
0020            IF(.NOT.((MOD(ITM4,KBLK).EQ.1).AND.(BUFCNT.GE.BUFTHR)))GO TO 7   11.6
0021                    GAPPED=YES                                               11.6
0022                    BTASAV=BETA(IBLK2)                                       11.6
0023                    BETA(IBLK2)=1.                                           11.6
0024                    DO 6 IX2=1,MXQM                                          11.6
0025                            DO 5 IX3=1,2                                     11.6
0026                                    THRSAV(IX2,IX3)=THR(IX2,IX3)             11.6
0027                                    THR(IX2,IX3)=1.E30                       11.6
0028    5                       CONTINUE                                         11.6
0029    6               CONTINUE                                                 11.6
0030                    THR(1,1)=-1.E30                                          11.6
0031            GO TO 7                                                          11.6
0032    7       CONTINUE                                                         11.6
0033            V(ITM6)=FLOAT(S(ITM3))-ALPHA*BETA(IBLK2)*
```

```
                1 SHAT(MODB((ITM3-T(IBLK2)),IBUF1))
##34             CALL INLOOP(V(ITM6),Q,VHAT(ITM7),BUFCNT)                    11.3A
      C         CALL INLOOP(V(ITM6),Q,VHAT(ITM7))                           1-1.3A
##35             CALL OUTBUF(BUFCNT,PREFIX,Q)                                11.2
##36             IF(ITM5.LE.0)GO TO 10
##37                     BTA=BETA(IBLK3)
##38                     IT1=T(IBLK3)
##39             GO TO 20
##40    10       CONTINUE
##41                     BTA=0.
##42                     IT1=1
##43             GO TO 20
##44    20       CONTINUE
##45             SHAT(ITM7)=VHAT(ITM7)+ALPHA*BTA*SHAT(MODB((ITM7-IT1),IBUF1))
##46             IF(.NOT.((MOD(ITM4,KBLK).EQ.0).AND.(GAPPED.EQV.YES)))GO TO 27  11.6
##47                     GAPPED=NO                                           11.6
##48                     BETA(IBLK2)=BTASAV                                  11.6
##49                     DO 26 IX2=1,MXQM                                    11.6
##50                             DO 25 IX3=1,2                               11.6
##51                                     THR(IX2,IX3)=THRSAV(IX2,IX3)        11.6
##52    25                       CONTINUE                                    11.6
##53    26               CONTINUE                                            11.6
##54             GO TO 27                                                    11.6
##55    27       CONTINUE                                                    11.6
##56             SHAT(ITM7)=AMAX1(MINVAL,(AMIN1(MAXVAL,SHAT(ITM7))))
##57             IF(.NOT.((ITM7.EQ.IBUF1).AND.(ITM5.GT.0)))GO TO 40
##58                     WRITE(LUNSH,30)(NINT(SHAT(IX1)),IX1=1,IBUF1)
##59    30               FORMAT(16I5)
##60             GO TO 40
##61    40       CONTINUE
##62             FS1=FLOAT(S(MODB(ITM5,IBUF2)))
##63             IF(BUFFIL.EQV.NO)GO TO 42                                   11.2
##64                     WRITE(LUNBUF,41)FS1,BUFCNT                          11.2
##65    41               FORMAT(F7.0,F10.1)                                  11.2
##66             GO TO 42                                                    11.2
##67    42       CONTINUE                                                    11.2
##68             ES=ES+FS1**2
##69             EN=EN+(FS1-SHAT(ITM7))**2
##70             IF(MOD(ITM5,1000).NE.0)GO TO 60
##71                     WRITE(LUNPRT,50)ITM5,ES,EN
##72    50               FORMAT(I7,2G12.4)
##73             GO TO 60
##74    60       CONTINUE
##75             RETURN
##76             END
```

```
FORTRAN IV-PLUS V02-51          15:59:43    15-APR-80           PAGE 1
MODB.FTN        /TR:BLOCKS/WR

000001              INTEGER FUNCTION MODB(INT1,INT2)
                C
                C       PURPOSE:  TO CALCULATE INT1 MOD INT2,
                C                  WHERE INT2=2**N, FOR SOME INTEGER N,
                C                  EXCEPT MODB(INT2,INT2)=INT2.
                C
                C       V.R     DATE            NAME            COMMENTS
                C       1.1     19-JUL-79       J.M. KRESSE     ORIGINATION
                C
                C--------DECLARATIONS
                C
000002              INTEGER INT1,INT2
                C
                C--------PROCEDURE
                C
000003              MODB=IAND(INT1-1,INT2-1)+1
000004              RETURN
000005              END
```

```
0001            SUBROUTINE INLOOP(V,Q,VHAT,BUFCNT)                              11.3A
      C         SUBROUTINE INLOOP(V,Q,VHAT)                                     1-1.3A
      C
      C         PURPOSE:  TO SIMULATE THE INLOOP SYSTEM.
      C
      C         V.R     DATE            NAME            COMMENTS
      C         0.1     16-JUL-79       J.M. KRESSE     EXISTING SOFTWARE
      C         1.0     17-JUL-79       J.M. KRESSE     BACKED UP
      C         1.3A    19-SEP-79       J.M. KRESSE     ADD BUFFER CONTROL
      C
      C         INLOOP TAKES THE REDUCED SPEECH, V, AT TIME N, AND RETURNS THE
      C         QUANTIZER LEVEL, Q, AND THE ESTIMATE OF THE REDUCED SPEECH,
      C         VHAT, FOR THE TIME N-L+1 (A DELAY OF L-1 UNITS).
      C
0002            PARAMETER MXL=10
0003            INTEGER QMAX,Q
0004            REAL BUFCNT                                                     11.3A
0005            COMMON /PARAM/M,L,N,QMAX
0006            COMMON /INBUF/VN(MXL)
0007            COMMON /NNODEO/NNODEO
0008            COMMON /NNODEN/NNODEN
      C
      C         SAVE V IN VN
      C
0009            IF(L.LE.1)GO TO 10
0010                    DO 20 IBCK=1,L-1
0011                            VN(L-IBCK+1)=VN(L-IBCK)
0012    20              CONTINUE
0013    10      CONTINUE
0014            VN(1)=V
      C
      C         NUMBER OF NEW NODES=NUMBER OF OLD NODES*NUMBER OF QUANTIZER
      C         LEVELS
      C
0015            NNODEN=NNODEO*QMAX
      C
      C         GROW NEW NODES
      C
0016            CALL GROW4(BUFCNT)                                              11.3A
      C         CALL GROW3                                                      1-1.3A
      C
      C         RANK NEW NODES
      C
0017            CALL RANKS3
```

```
C
C       ANNOUNCE DECISION, COLLECT STATISTICS, AND PRUNE THE TREE
C
      CALL PRUNE3(Q,VHAT)
      RETURN
      END
```

```
FORTRAN IV-PLUS V02-51          15:37:09    15-APR-80          PAGE 1
GROW4.FTN       /TR:BLOCKS/WR

0001            SUBROUTINE GROW4(BUFCNT)                                    11.3A
        C       SUBROUTINE GROW3                                            1-1.3A
        C
        C       PURPOSE:  TO GROW THE NEW NODES.
        C
        C       V.R     DATE            NAME            COMMENTS
        C       0.1     16-JUL-79       J.M. KRESSE     EXISTING SOFTWARE
        C       1.0     17-JUL-79       J.M. KRESSE     BACKED UP
        C       1.3A    19-SEP-79       J.M. KRESSE     ADD BUFFER CONTROL
        C
        C       SUBROUTINE GROW GROWS QMAX NEW NODES FROM EACH OLD NODE.
        C
0002            PARAMETER MXQM=11,MXN=8,MXM=1,MXL=10,MXLN=10
0003            INTEGER QMAX,QN,QO
0004            REAL BUFCNT,BUFFAC,ENORM,T                                  11.3A
0005            COMMON /PARAM/M,L,N,QMAX,MXVHN,DELTA,ALPH,EAASND,EAASNM,G
0006            COMMON /F/F(MXQM)/B/B(MXN)/ALPHA/ALPHA(MXQM)
0007            COMMON /NNODEO/NNODEO/VHATO/VHATO(MXM,MXLN-1)/QO/QO(MXM,MXL-1)
0008            COMMON /EAASO/EAASO(MXM)
0009            COMMON /AO/AO(MXM,MXN)/SIGMO/SIGMO(MXM)
0010            COMMON /NNODEN/NNODEN/CN/CN(MXQM*MXM)/VHATN/VHATN(MXQM*MXM,MXLN)
0011            COMMON /QN/QN(MXQM*MXM,MXL)
0012            COMMON /EAASN/EAASN(MXQM*MXM)/AN/AN(MXQM*MXM,MXN)
0013            COMMON /SIGMN/SIGMN(MXQM*MXM)
0014            COMMON /INBUF/VN(MXL)
0015            COMMON /T/T(MXQM,2)                                         11.3A
        C       WRITE(6,100)
        C 100   FORMAT('0NEW NODES:'//3X,'Q1 2',3X,'VHAT1',7X,'VHAT2',9X,'CN',
        C       18X,'EAASN',7X,'SIGMO',8X,'AO1',9X,'AO2',8X,'SIGMN',8X,'AN1',
        C       29X,'AN2'//)
        C
        C       DO FOR EACH OLD NODE
        C
0016            DO 10 INO=1,NNODEO
        C
        C               GROW QMAX NEW NODES
        C
0017                    DO 20 IQ=1,QMAX
        C
        C                       COMPUTE NEW NODE INDEX
        C
0018                            INN=(INO-1)*QMAX+IQ
        C
        C                       COMPUTE THE ESTIMATE OF THE ERROR
        C
0019                            EHAT=F(IQ)*SIGMO(INO)
```

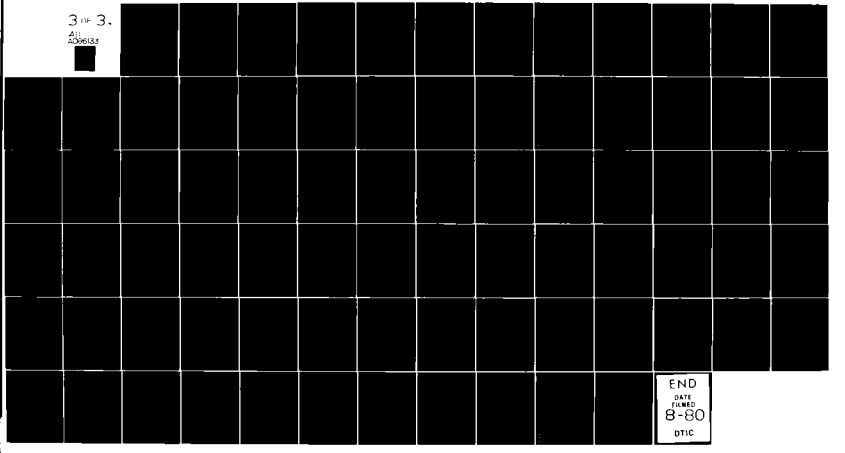
```
FORTRAN IV-PLUS V02-51          15:37:09    15-APR-80          PAGE 2
GROW4.FTN       /TR:BLOCKS/WR


        C
        C                       COMPUTE ESTIMATE OF REDUCED SPEECH
        C
0020                            VHATN(INN,1)=EHAT
0021                            DO 30 ICOEF=1,N
0022                                    VHATN(INN,1)=VHATN(INN,1)+AO(INO,ICOEF)*
              1                         VHATO(INO,ICOEF)
0023    30                      CONTINUE
        C
        C                       UPDATE Q
        C
0024                            QN(INN,1)=IQ
        C                       BRING FORWARD OTHER VHAT, Q FROM OLD
        C                       NODE
        C
0025                            DO 80 ITBK2=1,MXVHN
0026                                    VHATN(INN,ITBK2+1)=VHATO(INO,ITBK2)
0027    80                      CONTINUE
0028                            IF(L.LE.1)GO TO 70
0029                                    DO 40 ITBAK=1,L-1
0030                                            QN(INN,ITBAK+1)=QO(INO,ITBAK)
0031    40                              CONTINUE
0032    70                      CONTINUE
        D                       PRINT 90,INN,VHATN(INN,1),VHATN(INN,L),
        D             1         QN(INN,1),QN(INN,L)
        D  90                   FORMAT(I7,2G15.7,2I7)
        C
        C                       COMPUTE CRITERION
        C
0033                            CALL BUFCON(BUFCNT,BUFFAC)                      11.3A
0034                            BUFFAC=1.                                       11.6
0035                            ENORM=(VN(1)-VHATN(INN,1)+EHAT)/(SIGMO(INO)*    11.3A
              1                  BUFFAC)                                       11.3A
0036                            IF(.NOT.((T(IQ,1).LT.ENORM).AND.(ENORM.LE.      11.3A
              1                  T(IQ,2))))GO TO 75                            11.3A
0037                                    CN(INN)=0.                              11.3A
0038                            GO TO 76                                        11.3A
0039    75                      CONTINUE                                        11.3A
0040                                    CN(INN)=-1.                             11.3A
0041                            GO TO 76                                        11.3A
0042    76                      CONTINUE                                        11.3A
        C                       CN(INN)=0.                                      1-1.3A
        C                       DO 50 ITBK1=1,L                                 1-1.3A
        C                               CN(INN)=CN(INN)-(VN(ITBK1)-VHATN(INN,   1-1.3A
        C             1                 ITBK1))**2                            1-1.3A
        C  50                   CONTINUE                                        1-1.3A
```

```
C
C                       COMPUTE EXPONENTIAL AVERAGE OF ABSOLUTE VALUE OF
C                       VHAT
C
0043                    EAASN(INN)=(1.-ALPH)*ABS(VHATN(INN,1))+ALPH*
     1                  EAASO(INO)
C
C                       UPDATE PREDICTOR COEFFICIENTS
C
0044                    DO 60 ICOF1=1,N
0045                            CORRFA=(G*VHATN(INN,ICOF1+1)*EHAT)/
     1                          (EAASN(INN)+EAASNM)**2
0046                            AN(INN,ICOF1)=DELTA*B(ICOF1)+(1.-DELTA)*
     1                          (AO(INO,ICOF1)+CORRFA)
0047    60                      CONTINUE
C
C                       UPDATE SIGMA
C
0048                    SIGMN(INN)=AMAX1(ALPHA(IQ)*SIGMO(INO),
     1                  (EAASN(INN)+EAASNM)/EAASND)
C                       WRITE(6,110)INN,QN(INN,1),QN(INN,2),
C       1               VHATN(INN,1),VHATN(INN,2),CN(INN),EAASN(INN),
C       2               SIGMO(INO),AO(INO,1),AO(INO,2),SIGMN(INN),
C       3               AN(INN,1),AN(INN,2)
C 110                   FORMAT(I3,2I2,10G12.4)
0049    20              CONTINUE
0050    10      CONTINUE
0051            RETURN
0052            END
```

AD-A086 133 NOTRE DAME UNIV IN DEPT OF ELECTRICAL ENGINEERING F/G 17/2
DESIGN AND IMPLEMENTATION OF A SPEECH CODING ALGORITHM AT 9600 --ETC(U)
APR 80 J L MELSA, D L COHN, A ARORA DCA100-79-C-0005

UNCLASSIFIED NL

3 OF 3
AD A086133



END
DATE FILMED
8-80
DTIC

```
FORTRAN IV-PLUS V02-51          15:37:35    15-APR-80           PAGE 1
BUFCON.FTN      /TR:BLOCKS/WR

0001            SUBROUTINE BUFCON(BUFCNT,BUFFAC)
        C
        C       PURPOSE:  TO COMPUTE THE BUFFER CONTROL FACTOR.
        C
        C       V.R     DATE            NAME            COMMENTS
        C       1.3A    19-SEP-79       J.M. KRESSE     ADD BUFFER CONTROL
        C       1.4     26-SEP-79       J.M. KRESSE     ADD BUFFER OCCUPANCY
        C
        C-------DECLARATIONS
        C
0002            PARAMETER NRANGE=10
0003            INTEGER BUFOCC                                              11.4
0004            INTEGER IX
0005            REAL BFFCAR,BUFCNT,BUFFAC,RANGE
0006            COMMON /RANGE/ RANGE(NRANGE+1)
0007            COMMON /BFFCAR/ BFFCAR(NRANGE)
0008            COMMON /BUFOCC/ BUFOCC(NRANGE)                              11.4
        C
        C-------PROCEDURE
        C
0009            DO 10 IX=1,NRANGE
0010                    IF(.NOT.((RANGE(IX).LE.BUFCNT).AND.(BUFCNT.LT.      11.4
        1        RANGE(IX+1))))GO TO 5                                      11.4
0011                            BUFFAC=BFFCAR(IX)                           11.4
0012                            BUFOCC(IX)=BUFOCC(IX)+1                     11.4
0013                    GO TO 5                                             11.4
0014    5               CONTINUE                                            11.4
        C               IF((RANGE(IX).LE.BUFCNT).AND.(BUFCNT.LE.RANGE(IX+1)))  1-1.4
        C       1        BUFFAC=BFFCAR(IX)                                  1-1.4
0015    10      CONTINUE
0016            RETURN
0017            END
```

```
FORTRAN IV-PLUS V02-51          15:37:46    15-APR-80          PAGE 1
RANKS4.FTN      /TR:BLOCKS/WR

000001              SUBROUTINE RANKS3
                C
                C   PURPOSE:  TO RANK THE NEW NODES.
                C
                C   V.R     DATE            NAME            COMMENTS
                C   0.1     16-JUL-79       J.M. KRESSE     EXISTING SOFTWARE
                C   1.0     17-JUL-79       J.M. KRESSE     BACKED UP
                C
                C   SUBROUTINE RANKS RANKS THE NEW NODES IN DESCENDING ORDER OF THE
                C   CRITERION.  THE ARRAY RANK STORES NODE INDICES:  E.G., RANK(1)=
                C   NODE INDEX OF NEW NODE WITH HIGHEST VALUE OF CRITERION.
                C
000002              PARAMETER MXQM=11,MXM=1
000003              INTEGER RANK,TEMP
000004              COMMON /NNODEN/NNODEN/CN/CN(MXQM*MXM)
000005              COMMON /RANK1/RANK(MXQM*MXM)
                C
                C   INITIALIZE RANKINGS
                C
000006              DO 10 NINDX=1,NNODEN
                C
                C           INITIAL RANK IS NODE INDEX
                C
000007                      RANK(NINDX)=NINDX
000008     10       CONTINUE
```

```
FORTRAN IV-PLUS V02-51          15:37:46    15-APR-80          PAGE 2
RANKS4.FTN      /TR:BLOCKS/WR


        C
        C       PERFORM BUBBLE SORT
        C
        C       DO (NNODEN-1) PASSES
        C
0009            DO 20 NPASS=1,NNODEN-1
        C
        C               DO (NNODEN-NPASS) COMPARISONS ON THE PASS
        C
0010                    DO 30 NCOMP=1,NNODEN-NPASS
        C
        C                       THE COMPARISON:
        C
0011                            IF(CN(RANK(NCOMP)).GE.CN(RANK(NCOMP+1)))GO TO 40
        C
        C                               ORDER REVERSED: REVERSE THEM
        C
0012                            TEMP=RANK(NCOMP)
0013                            RANK(NCOMP)=RANK(NCOMP+1)
0014                            RANK(NCOMP+1)=TEMP
0015    40                      CONTINUE
0016    30              CONTINUE
0017    20      CONTINUE
        C       WRITE(6,50)(RANK(NINDX1),NINDX1=1,NNODEN)
        C 50    FORMAT(//'0THE RANKING:'//(NNODEN)I3//)
0018            RETURN
0019            END
```

```
FORTRAN IV-PLUS V02-51          15:38:00    15-APR-80          PAGE 1
PRUNE5.FTN      /TR:BLOCKS/WR

0001            SUBROUTINE PRUNE3(Q,VHAT)
                C
                C       PURPOSE:  TO PRUNE THE NEW NODES.
                C
                C       V.R     DATE            NAME            COMMENTS
                C       0.1     16-JUL-79       J.M. KRESSE     EXISTING SOFTWARE
                C       1.0     17-JUL-79       J.M. KRESSE     BACKED UP
                C
                C       SUBROUTINE PRUNE ANNOUNCES THE QUANTIZER LEVEL (AND THE
                C       ASSOCIATED VHAT), COLLECTS STATISTICS, AND PRUNES THE TREE IN
                C       PREPARATION FOR THE NEXT SPEECH SAMPLE.
                C
0002            PARAMETER MXL=10,MXLN=10,MXM=1,MXN=8,MXQM=11
0003            PARAMETER HINOIS=.FALSE.
0004            PARAMETER LONOIS=.TRUE.
0005            BYTE QPLOT
0006            INTEGER Q,QLVL,QMAX,QN,RANK
0007            LOGICAL LEVEL
0008            REAL LOSNR,NOISE
0009            COMMON /AO/ AO(MXM,MXN)
0010            COMMON /CN/ CN(MXQM*MXM)
0011            COMMON /INBUF/ VN(MXL)
0012            COMMON /MADAPT/ FACTOR,HISNR,LOSNR
0013            COMMON /MEFDIS/ MEFDIS(MXM)
0014            COMMON /MVARY/ LEVEL,M1,NLONO,NHINO
0015            COMMON /NNODEN/ NNODEN
0016            COMMON /NNODEO/ NNODEO
0017            COMMON /PARAM/ M,L,N,QMAX,MXVHN
0018            COMMON /QLVL/ QLVL(MXQM)
0019            COMMON /QN/ QN(MXQM*MXM,MXL)
0020            COMMON /QPLOT/ QPLOT
0021            COMMON /RANK1/ RANK(MXQM*MXM)
0022            COMMON /SIGMO/ SIGMO(MXM)
0023            COMMON /STATS/ NSAMP,VSQR,NOISE
0024            COMMON /VHATN/ VHATN(MXQM*MXM,MXLN)
                C
                C       ANNOUNCE DECISIONS
                C
0025            Q=QN(RANK(1),L)
0026            VHAT=VHATN(RANK(1),L)
                C
                C       COLLECT SOME STATISTICS
                C
                C       INCREMENT SAMPLE COUNTER
                C
0027            NSAMP=NSAMP+1
0028            IF(MOD(NSAMP,1000).EQ.0)WRITE(6,10)SIGMO(1),(AO(1,IX1),IX1=1,N),
                1-CN(RANK(1)),MEFDIS
0029    10      FORMAT(<N+2>G12.4/<MXM>I6)
                C
                C       COLLECT SNR DATA
                C
0030            VSQR=VSQR+VN(L)**2
0031            NOISE=NOISE+(VN(L)-VHAT)**2
                C
                C       COLLECT QUANTIZER LEVEL STATISTICS
```

```
FORTRAN IV-PLUS V02-51          15:38:00    15-APR-80          PAGE 2
PRUNE5.FTN      /TR:BLOCKS/WR

      C
00032           QLVL(Q)=QLVL(Q)+1
```

```
C
C       COLLECT EFFECTIVE L STATISTICS
C
      CALL LEFST
C
C       COLLECT EFFECTIVE M STATISTICS
C
      MEFDIS(NNODEO)=MEFDIS(NNODEO)+1
      IF(QPLOT.EQ.'Y')CALL MFILE
C
C       PRUNE THE TREE BY NOT SAVING INFORMATION FROM THE PRUNED NODES.
C
      IF(LEVEL.EQV.HINOIS)GO TO 20
          NLONO=NLONO+1
      GO TO 30
20    NHINO=NHINO+1
      GO TO 30
30    CONTINUE
      CALL SNRVAL(-CN(RANK(1)),SNR)
      IF(.NOT.((LEVEL.EQV.LONOIS).AND.(SNR.LT.LOSNR)))GO TO 40
          LEVEL=HINOIS
          M=M1
      GO TO 40
40    CONTINUE
      IF(.NOT.((LEVEL.EQV.HINOIS).AND.(SNR.GT.HISNR)))GO TO 50
          LEVEL=LONOIS
          M=1
      GO TO 50
50    CONTINUE
      NNODEO=0
      DO 70 INN=1,NNODEN
          IF(.NOT.
     1     ((QN(RANK(INN),L).EQ.QN(RANK(1),L))
     2     .AND.(NNODEO.LT.M)
     3     .AND.((-CN(RANK(INN))).LE.FACTOR*(-CN(RANK(1))))))
     4     GO TO 60
                NNODEO=NNODEO+1
                CALL SAVE(INN)
          CONTINUE
60    CONTINUE
70    RETURN
      END
```

```
FORTRAN IV-PLUS V02-51          15:38:26    15-APR-80          PAGE 1
LEFST.FTN       /TR:BLOCKS/WR

0001            SUBROUTINE LEFST
                C
                C       PURPOSE:  TO COLLECT THE EFFECTIVE L STATISTICS.
                C
                C       V.R     DATE            NAME            COMMENTS
                C       0.1     16-JUL-79       J.M. KRESSE     EXISTING SOFTWARE
                C       1.0     17-JUL-79       J.M. KRESSE     BACKED UP
                C
                C       LEFST COLLECTS EFFECTIVE L STATISTICS
                C
0002            PARAMETER MXL=10,MXM=1,MXQM=11
0003            PARAMETER DIFF=.FALSE.,SAME=.TRUE.
0004            INTEGER QN
0005            LOGICAL COND
0006            COMMON /LEFDIS/ LEFDIS(MXL)
0007            COMMON /NNODEN/ NNODEN
0008            COMMON /PARAM/ M,L
0009            COMMON /QN/ QN(MXQM*MXM,MXL)
                C
                C
                C
0010            COND=SAME
0011            LEVEL=L
                C
                C       DO UNTIL (DIFFERENCE FOUND)
                C
0012    10      CONTINUE
0013                    NODE=2
                C
                C               DO UNTIL ((OUT OF NODES) OR (DIFFERENCE FOUND))
                C
0014    20              CONTINUE
0015                            IF(QN(NODE,LEVEL).NE.QN(1,LEVEL))COND=DIFF
0016                            IF((NODE.EQ.NNODEN).OR.(COND.EQV.DIFF))GO TO 30
0017                            NODE=NODE+1
0018                    GO TO 20
0019    30              CONTINUE
0020                    IF(COND.EQV.DIFF)GO TO 40
0021                    LEVEL=LEVEL-1
0022            GO TO 10
0023    40      CONTINUE
0024            LEFDIS(LEVEL)=LEFDIS(LEVEL)+1
0025            RETURN
0026            END
```

```
FORTRAN IV-PLUS V02-51          15:38:38    15-APR-80          PAGE 1
MFILE.FTN       /TR:BLOCKS/WR

0001            SUBROUTINE MFILE
        C
        C       PURPOSE:  TO COLLECT THE EFFECTIVE M STATISTICS AND PUT THEM OUT
        C                  ON LOGICAL UNIT 4.
        C
        C       V.R     DATE            NAME            COMMENTS
        C       0.1     16-JUL-79       J.M. KRESSE     EXISTING SOFTWARE
        C       1.0     17-JUL-79       J.M. KRESSE     BACKED UP
        C
        C-------DECLARATIONS
        C
0002            PARAMETER BUFFL=16
0003            COMMON /ICNT/ ICNT
0004            COMMON /MBUFF/ MBUFF(BUFFL)
0005            COMMON /NNODEO/ NNODEO
        C
        C-------PROCEDURE
        C
0006            ICNT=ICNT+1
0007            MBUFF(ICNT)=NNODEO
0008            IF(ICNT.NE.BUFFL)GO TO 20
0009                    ICNT=0
0010                    WRITE(4,10)MBUFF
0011    10              FORMAT(<BUFFL>I5)
0012    20      CONTINUE
0013            RETURN
0014            END
```

```
FORTRAN IV-PLUS V02-51          15:38:49    15-APR-80          PAGE 1
SNRVAL.FTN      /TR:BLOCKS/WR

0001            SUBROUTINE SNRVAL(NOISE,SNR)
                C
                C       PURPOSE:  TO CALCULATE THE SNR CORRESPONDING TO NOISE.
                C
                C       V.R     DATE            NAME            COMMENTS
                C       0.1     16-JUL-79       J.M. KRESSE     EXISTING SOFTWARE
                C       1.0     17-JUL-79       J.M. KRESSE     BACKED UP
                C
                C-------DECLARATIONS
                C
0002            PARAMETER MXL=10
0003            PARAMETER MAXVAL=1.E30,MINVAL=-1.E30
0004            REAL NOISE
0005            COMMON /INBUF/ VN(MXL)
0006            COMMON /PARAM/ M,L
                C
                C-------PROCEDURE
                C
0007            IF(NOISE.NE.0.)GO TO 10
0008                    SNR=MAXVAL
0009            GO TO 50
0010    10      CONTINUE
0011                    SIG=0.
0012                    DO 20 IX1=1,L
0013                            SIG=SIG+VN(IX1)**2
0014    20              CONTINUE
0015                    IF(SIG.NE.0.)GO TO 30
0016                            SNR=MINVAL
0017                    GO TO 40
0018    30              CONTINUE
0019                            SNR=10.*ALOG10(SIG/NOISE)
0020                            SNR=AMAX1(SNR,MINVAL)
0021                            SNR=AMIN1(SNR,MAXVAL)
0022                    GO TO 40
0023    40              CONTINUE
0024            GO TO 50
0025    50      CONTINUE
0026            RETURN
0027            END
```

```
0001            SUBROUTINE OUTBUF(BUFCNT,PREFIX,Q)
                C
                C       PURPOSE:  TO COMPUTE THE SIZE OF THE BUFFER AFTER EACH SAMPLE.
                C
                C       V.R     DATE            NAME            COMMENTS
                C       1.2     22-JUL-79       J.M. KRESSE     ORIGINATION
                C       1.3A    19-SEP-79       J.M. KRESSE     ADD BUFFER CONTROL
                C       1.4     08-OCT-79       J.M. KRESSE     NEW SOURCE CODE
                C
                C-------DECLARATIONS
                C
                C       PARAMETER RUNLEN=14,RNCOLN=4.                           1-1.8
0002            PARAMETER RUNLEN=10,RNCOLN=-7.                                  11.8
0003            PARAMETER R2NLEN=14,R2COLN=4.                                   11.9
0004            PARAMETER QMAX=11,XMTCNT=1.35                                   11.3A
                C       PARAMETER QMAX=11,XMTCNT=1.5                            1-1.3A
0005            INTEGER PREFIX,Q
0006            INTEGER R2CNT                                                   11.9
0007            REAL BITCNT(QMAX),BUFCNT                                        11.4
                C       REAL BITCNT(QMAX,0:1),BUFCNT                            1-1.4
                C
                C-------DEFINITIONS
                C
0008            DATA BITCNT /0.,3.,3.,4.,4.,6.,6.,7.,7.,7.,7./                  11.4
                C       DATA BITCNT /.5,3.,3.,6.,6.,10.,8.,10.,10.,11.,10.,     1-1.4
                C       1 .5,2.5,3.5,6.5,6.5,9.5,7.5,10.5,9.5,10.5,10.5/        1-1.4
                C
                C-------PROCEDURE
                C
0009            IF(Q.EQ.1)GO TO 10                                              11.4
                C               BUFCNT=BUFCNT+PREFIX+BITCNT(Q)                  1-1.9
0010            R2CNT=PREFIX/R2NLEN                                             11.9
0011            BUFCNT=BUFCNT+FLOAT(R2CNT)*R2COLN+FLOAT(PREFIX-                 11.9
                1        R2CNT*R2NLEN)+BITCNT(Q)                                11.9
0012            PREFIX=0                                                        11.4
0013            GO TO 30                                                        11.4
0014    10      CONTINUE                                                        11.4
0015            PREFIX=PREFIX+1                                                 11.4
0016            IF(PREFIX.NE.RUNLEN)GO TO 20                                    11.4
0017                    BUFCNT=BUFCNT+RNCOLN                                    11.4
0018                    PREFIX=0                                                11.4
0019            GO TO 20                                                        11.4
0020    20      CONTINUE                                                        11.4
0021            GO TO 30                                                        11.4
0022    30      CONTINUE                                                        11.4
0023            BUFCNT=AMAX1(0.,BUFCNT-XMTCNT)                                  11.4
                C       BUFCNT=AMAX1(0.,BUFCNT+BITCNT(Q,PREFIX)-XMTCNT)         11.3A
                C       BUFCNT=BUFCNT+BITCNT(Q,PREFIX)-XMTCNT                   1-1.3A
                C       IF(.NOT.((PREFIX.EQ.0).AND.(Q.EQ.1)))GO TO 10
                C               PREFIX=1
                C       GO TO 20
                C 10    CONTINUE
                C               PREFIX=0
                C       GO TO 20
                C 20    CONTINUE
0024            RETURN
```

```
0025          END
```

```
0001            SUBROUTINE INEND
                C
                C       PURPOSE:  TO COLLECT AND OUTPUT STATISTICAL DATA.
                C
                C       V.R     DATE            NAME            COMMENTS
                C       0.1     16-JUL-79       J.M. KRESSE     EXISTING SOFTWARE
                C       1.0     17-JUL-79       J.M. KRESSE     BACKED UP
                C       1.4     26-SEP-79       J.M. KRESSE     ADD BUFFER OCCUPANCY
                C
                C       SUBROUTINE INEND GATHERS AND OUTPUTS STATISTICAL DATA
                C
0002            PARAMETER MXL=10,MXM=1,MXQM=11
0003            PARAMETER NRANGE=10                                          11.4
0004            INTEGER BUFOCC                                               11.4
0005            INTEGER QLVL,QMAX
0006            LOGICAL LEVEL
0007            REAL NOISE
0008            COMMON /BUFOCC/ BUFOCC(NRANGE)                               11.4
0009            COMMON /LEFDIS/ LEFDIS(MXL)
0010            COMMON /MEFDIS/ MEFDIS(MXM)
0011            COMMON /MVARY/ LEVEL,M1,NLONO,NHINO
0012            COMMON /PARAM/ M,L,N,QMAX
0013            COMMON /QLVL/ QLVL(MXQM)
0014            COMMON /STATS/ NSAMP,VSQR,NOISE
                C
                C       OUTPUT THE NUMBER OF SAMPLES HANDLED BY THE TREE SEARCHING
                C       ALGORITHM.
                C
0015            WRITE(6,10)NSAMP
0016    10      FORMAT('0# OF SAMPLES (TREE SEARCHING)= ',I5)
0017            WRITE(6,20)NLONO,NHINO
0018    20      FORMAT(' # OF SAMPLES (M=1)= ',I5,' # OF SAMPLES (M=M)= ',I5)
                C
                C       OUTPUT SNR FOR TREE SEARCHING
                C
0019            WRITE(6,30)10.*ALOG10(VSQR/NOISE)
0020    30      FORMAT(' SNR FOR TREE SEARCHING= ',G12.4,' DB ')
```

```
C
C       OUTPUT QUANTIZER LEVEL FREQUENCY DISTRIBUTION AND ENTROPY
C
      ALOG2=ALOG(2.)
      ENTRPY=0.
      WRITE(6,40)
   40 FORMAT('0QUANTIZER LEVEL FREQUENCY DISTRIBUTION: '/
     1' LEVEL NUMBER',5X,'P',7X,'-P*LOG(P)')
      DO 80 IQ1=1,QMAX
         FREQQ=FLOAT(QLVL(IQ1))/FLOAT(NSAMP)
         IF(FREQQ.EQ.0.)GO TO 50
            ANLN=-FREQQ*ALOG(FREQQ)/ALOG2
         GO TO 60
   50       ANLN=0.
         GO TO 60
   60    CONTINUE
         ENTRPY=ENTRPY+ANLN
         WRITE(6,70)IQ1,QLVL(IQ1),FREQQ,ANLN
   70    FORMAT(I4,4X,I5,2G12.4)
   80 CONTINUE
      WRITE(6,90)ENTRPY
   90 FORMAT('0CORRESPONDING ENTROPY: ',G12.4,' BITS/SAMPLE ')
      WRITE(6,100)
  100 FORMAT('0L(EFF)',3X,'X',5X,'*')
      WRITE(6,110)(IX1,(100.*FLOAT(LEFDIS(IX1)))/FLOAT(NSAMP),
     1 LEFDIS(IX1),IX1=1,L)
  110 FORMAT(I5,F8.2,I6)
      WRITE(6,120)
  120 FORMAT('0M(EFF)',3X,'X',5X,'*')
      WRITE(6,130)(IX2,(100.*FLOAT(MEFDIS(IX2)))/FLOAT(NSAMP),
     1 MEFDIS(IX2),IX2=1,M1)
  130 FORMAT(I5,F8.2,I6)
      WRITE(6,*)'BUFFER OCCUPANCY:',BUFOCC                          11.4
      RETURN
      END
```

```
0001            SUBROUTINE SAVE(NRANK)
      C
      C         PURPOSE:  TO SAVE THE INFORMATION FROM THE NODE RANKED NRANK.
      C
      C         V.R     DATE            NAME            COMMENTS
      C         0.1     16-JUL-79       J.M. KRESSE     EXISTING SOFTWARE
      C         1.0     17-JUL-79       J.M. KRESSE     BACKED UP
      C
      C         SAVE SAVES THE INFORMATION FROM THE NODE RANKED NRANK.
      C
0002            PARAMETER MXL=10,MXLN=10,MXM=1,MXN=8,MXQM=11
0003            INTEGER QMAX,QN,QO,RANK
0004            COMMON /AN/ AN(MXQM*MXM,MXN)
0005            COMMON /AO/ AO(MXM,MXN)
0006            COMMON /EAASN/ EAASN(MXQM*MXM)
0007            COMMON /EAASO/ EAASO(MXM)
0008            COMMON /NNODEO/ NNODEO
0009            COMMON /PARAM/ M,L,N,QMAX,MXVHN
0010            COMMON /QN/ QN(MXQM*MXM,MXL)
0011            COMMON /QO/ QO(MXM,MXL-1)
0012            COMMON /RANK1/ RANK(MXQM*MXM)
0013            COMMON /SIGMN/ SIGMN(MXQM*MXM)
0014            COMMON /SIGMO/ SIGMO(MXM)
0015            COMMON /VHATN/ VHATN(MXQM*MXM,MXLN)
0016            COMMON /VHATO/ VHATO(MXM,MXLN-1)
      C
      C
      C
0017            DO 10 ICOEF=1,N
0018                    AO(NNODEO,ICOEF)=AN(RANK(NRANK),ICOEF)
0019    10      CONTINUE
0020            SIGMO(NNODEO)=SIGMN(RANK(NRANK))
0021            EAASO(NNODEO)=EAASN(RANK(NRANK))
0022            DO 20 IBCK1=1,MXVHN
0023                    VHATO(NNODEO,IBCK1)=VHATN(RANK(NRANK),IBCK1)
0024    20      CONTINUE
0025            IF(L.LE.1)GO TO 40
0026                    DO 30 IBCK=1,L-1
0027                            QO(NNODEO,IBCK)=QN(RANK(NRANK),IBCK)
0028    30              CONTINUE
0029    40      CONTINUE
0030            RETURN
0031            END
```

APPENDIX B
SEGMENTED SNR PLOTS

B.1. Introduction

A number of similar programs were developed to aid in the analysis of the performance of PARC. The purpose of the programs were to generate plots, called SNRSIG, which indicate graphically the short-time performance of the system versus the short-time signal level. This information proved useful, as it appears that the average performance of the systems over short periods of time is more indicative than the overall average performance.

There are three groups of programs, with two programs in each group. The first program in each group performs the short-time analysis, and generates the data to be plotted. The second program then takes the data and generates the actual plot.

The first group, DBCALC and SNRSIG, deals with the short-time signal-to-noise ratio. This group is useful in analyzing the performance of the quantizer, especially problems like slope overload noise and granular noise. The second group, BFCALC and BFPLOT, deals with the average buffer length, and the third group, BFDIFF and BFDFPL, deals with the difference in the average buffer length. These programs are useful in analyzing the performance of the source coding and of the buffer control.

```
$
[redacted] [redacted] C[redacted]ND [redacted] TO[redacted]LD [redacted]ALC.[redacted]
$
DBCALC/CP/FP,DBCALC/NOSP=DBCALC,ASK1,OPENER,ASK2,GETSEM,DBC,PUTSEM,
CLOSER
/
ACTFIL=5
//
```

```
0001            PROGRAM DBCALC
        C
        C       PURPOSE:  TO CALCULATE THE LOCAL SIGNAL STRENGTH AND THE LOCAL
        C                  SNR FOR A SPEECH FILE AND AN SHAT FILE.
        C
        C       V.R     DATE            NAME            COMMENTS
        C       0.1     28-JUN-79       J.M. KRESSE     ORIGINATION
        C       0.2     12-JUL-79       J.M. KRESSE     TO ADD THE SNR HEADER
        C
        C-------DECLARATIONS
        C
0002            PARAMETER DIMENS=33,MAXBUF=16,MAXLEN=1000,MAXVAL=1.E30
0003            PARAMETER MINVAL=-1.E30,NO=.FALSE.,YES=.TRUE.
0004            BYTE DBFILE(DIMENS),SFILE(DIMENS),SHFILE(DIMENS)
0005            INTEGER BLKLEN,BPOINT,BUFLEN,LUNDB,LUNS,LUNSCR,LUNSH
0006            INTEGER S(MAXLEN),SBUF(MAXBUF),SHAT(MAXLEN),SHBUF(MAXBUF),SPOINT
0007            LOGICAL EOF,MORES
0008            REAL OVRSNR,SIGSTR,SNR,TOTSIG,TOTNOI
        C
        C-------INITIALIZATION
        C
0009            BPOINT=MAXBUF
0010            BUFLEN=MAXBUF
0011            EOF=NO
0012            MORES=YES
0013            TOTNOI=0.
0014            TOTSIG=0.
        C
        C-------PROCEDURE
        C
        C       ASK FOR FILENAMES
        C
0015            CALL ASK1(SFILE,SHFILE,DBFILE,DIMENS)
        C
        C       OPEN AND READY THE FILES
        C
0016            CALL OPENER(SFILE,SHFILE,DBFILE,DIMENS,LUNS,LUNSH,LUNDB,LUNSCR)
        C
        C       ASK FOR THE DESIRED CALCULATION BLOCK LENGTH.
        C
0017            CALL ASK2(MAXLEN,BLKLEN)
        C
        C       DO UNTIL (MORES.EQV.NO)
        C
0018    10      CONTINUE
        C
        C               GET S, SHAT
        C
0019                    CALL GETSEM(S,SHAT,SBUF,SHBUF,LUNS,LUNSH,BLKLEN,SPOINT,
        1        MAXBUF,BUFLEN,BPOINT,EOF,MORES)
        C
        C               IF (SPOINT.NE.0)
        C
0020                    IF(SPOINT.EQ.0)GO TO 20
        C
        C                       CALCULATE THE LOCAL SIGNAL STRENGTH AND SNR.
```

```
C
0021                        CALL DBC(S,SHAT,SPOINT,SIGSTR,SNR,TOTSIG,TOTNOI)
C
C                           PUT OUT THE SIGNAL STRENGTH AND SNR TO THE
C                            SCRATCHFILE.
C
0022                        CALL PUTSEM(SIGSTR,SNR,LUNSCR)
0023                  GO TO 20
0024     20           CONTINUE
0025           IF(MORES.EQV.YES)GO TO 10
0026           CONTINUE
C
C      CALCULATE THE OVERALL SNR.
C
0027           OVRSNR=AMIN1(MAXVAL,AMAX1(MINVAL,10.*ALOG10(TOTSIG/TOTNOI)))
C
C      WRITE THE OVERALL SNR HEADER ON THE DBFILE.
C
0028           WRITE(LUNDB,30)OVRSNR
0029     30    FORMAT(G12.4)
C
C      COPY THE DATA FROM THE SCRATCHFILE TO THE DBFILE.
C
0030           REWIND LUNSCR
0031     40    CONTINUE
0032                  READ(LUNSCR,*,END=60)SIGSTR,SNR
0033                  WRITE(LUNDB,50)SIGSTR,SNR
0034     50           FORMAT(2G12.4)
0035           GO TO 40
0036     60    CONTINUE
C
C      CLOSE THE FILES.
C
0037           CALL CLOSER(LUNS,LUNSH,LUNDB,LUNSCR)
0038           STOP
0039           END
```

```
0001            SUBROUTINE ASK1(SFILE,SHFILE,DBFILE,DIMENS)
                C
                C       PURPOSE: TO ASK FOR THE FILENAMES NEEDED
                C
                C       V.R     NAME            DATE            COMMENTS
                C       0.1     J.M. KRESSE     26-JUN-79       ORIGINATION
                C
                C-------DECLARATIONS
                C
0002            PARAMETER NUL=0
0003            INTEGER DIMENS
0004            BYTE DBFILE(DIMENS),SFILE(DIMENS),SHFILE(DIMENS)
                C
                C-------PROCEDURE
                C
0005            WRITE(5,10)
0006    10      FORMAT(' WHAT IS THE SPEECH FILENAME?')
0007            READ(5,20)SFILE
0008    20      FORMAT(<DIMENS>A1)
0009            WRITE(5,30)
0010    30      FORMAT(' WHAT IS THE SHAT FILENAME?')
0011            READ(5,40)SHFILE
0012    40      FORMAT(<DIMENS>A1)
0013            WRITE(5,50)
0014    50      FORMAT(' WHAT DO YOU WANT TO CALL THE OUTPUT FILE?')
0015            READ(5,60)DBFILE
0016    60      FORMAT(<DIMENS>A1)
                C
                C       SET END-OF-STRING MARKS
                C
0017            SFILE(DIMENS)=NUL
0018            SHFILE(DIMENS)=NUL
0019            DBFILE(DIMENS)=NUL
                C
0020            RETURN
0021            END
```

```
0001            SUBROUTINE OPENER(SFILE,SHFILE,DBFILE,DIMENS,
                1 LUNS,LUNSH,LUNDB,LUNSCR)
                C
                C       PURPOSE: TO OPEN AND READY THE FILES REQUESTED.
                C
                C       V.R     DATE            NAME            COMMENTS
                C       0.1     26-JUN-79       J.M. KRESSE     ORIGINATION
                C       0.2     12-JUL-79       J.M. KRESSE     TO ADD THE SNR HEADER
                C
                C-------DECLARATIONS
                C
0002            INTEGER DIMENS
0003            BYTE DBFILE(DIMENS),SFILE(DIMENS),SHFILE(DIMENS)
0004            INTEGER LUNDB,LUNS,LUNSCR,LUNSH
                C
                C-------INITIALIZATIONS
                C
0005            LUNS=1
0006            LUNSH=2
0007            LUNDB=3
0008            LUNSCR=4
                C
                C-------PROCEDURE
                C
0009            OPEN(UNIT=LUNS,NAME=SFILE,TYPE='OLD',READONLY,SHARED)
0010            OPEN(UNIT=LUNSH,NAME=SHFILE,TYPE='OLD',READONLY,SHARED)
0011            OPEN(UNIT=LUNDB,NAME=DBFILE,TYPE='NEW',CARRIAGECONTROL='LIST')
0012            OPEN(UNIT=LUNSCR,TYPE='SCRATCH',CARRIAGECONTROL='LIST')
0013            READ(LUNS,10)
0014            READ(LUNSH,10)
0015    10      FORMAT(80X)
0016            RETURN
0017            END
```

```
0001            SUBROUTINE ASK2(MAXLEN,BLKLEN)
                C
                C       PURPOSE: TO ASK FOR THE CALCULATION BLOCK LENGTH.
                C
                C       V.R     NAME            DATE            COMMENTS
                C       0.1     J.M. KRESSE     26-JUN-79       ORIGINATION
                C
                C-------DECLARATIONS
                C
0002            PARAMETER MINLEN=1,RATE=6.4
0003            INTEGER BLKLEN,MAXLEN
0004            REAL BLKTIM,MAXTIM,MINTIM
                C
                C-------FUNCTIONS
                C
0005            TIME(INT)=FLOAT(INT)/RATE
                C
                C-------PROCEDURE
                C
0006            MINTIM=TIME(MINLEN)
0007            MAXTIM=TIME(MAXLEN)
                C
                C       DO UNTIL A VALID BLOCK LENGTH IS INPUT
                C
0008    10      CONTINUE
0009                    WRITE(5,20)MINLEN,MAXLEN,MINTIM,MAXTIM
0010    20              FORMAT(' THE CALCULATION BLOCK LENGTH MUST BE BETWEEN ',
                1        I5,' AND ',I5,' SAMPLES,'/' INCLUSIVE (BETWEEN ',G11.4,
                2        ' AND ',G11.4,' MS)'/
                3        ' HOW MANY SAMPLES PER BLOCK DO YOU WANT?')
0011                    READ(5,*)BLKLEN
0012            IF((BLKLEN.LT.MINLEN).OR.(BLKLEN.GT.MAXLEN))GO TO 10
0013            CONTINUE
0014            BLKTIM=TIME(BLKLEN)
0015            WRITE(5,30)BLKTIM
0016    30      FORMAT(' THE BLOCK IS ',G11.4,' MS LONG.')
0017            RETURN
0018            END
```

```
0001            SUBROUTINE GETSEM(S,SHAT,SBUF,SHBUF,LUNS,LUNSH,BLKLEN,SPOINT,
                1 MAXBUF,BUFLEN,BPOINT,EOF,MORES)
                C
                C       PURPOSE: TO GET A BLOCK OF S, SHAT SAMPLES
                C
                C
                C       V.R     NAME            DATE            COMMENTS
                C       0.1     J.M. KRESSE     26-JUN-79       ORIGINATION
                C
                C-------DECLARATIONS
                C
0002            PARAMETER DIGITS=5,NO=.FALSE.,YES=.TRUE.
0003            INTEGER BLKLEN,BPOINT,BUFLEN,COUNT,IX,MAXBUF,S(BLKLEN)
0004            INTEGER SBUF(MAXBUF),SHAT(BLKLEN),SHBUF(MAXBUF),SPOINT
0005            LOGICAL EOF,MORES
                C
                C-------INITIALIZATION
                C
0006            SPOINT=0
                C
                C-------PROCEDURE
                C
                C       DO WHILE ((SPOINT.LT.BLKLEN).AND.(MORES.EQV.YES))
                C
0007    10      IF(.NOT.((SPOINT.LT.BLKLEN).AND.(MORES.EQV.YES)))GO TO 110
                C
                C               DO WHILE ((BPOINT.LT.BUFLEN).AND.(SPOINT.LT.BLKLEN))
                C
0008    20              IF(.NOT.((BPOINT.LT.BUFLEN).AND.(SPOINT.LT.BLKLEN)))
                1        GO TO 30
0009                            BPOINT=BPOINT+1
0010                            SPOINT=SPOINT+1
0011                            S(SPOINT)=SBUF(BPOINT)
0012                            SHAT(SPOINT)=SHBUF(BPOINT)
0013                    GO TO 20
0014    30              CONTINUE
                C
                C               CASE:((BPOINT.EQ.BUFLEN).AND.(EOF.EQV.NO))
                C
0015                    IF(.NOT.((BPOINT.EQ.BUFLEN).AND.(EOF.EQV.NO)))GO TO 80
0016                            READ(LUNS,50,END=40)COUNT,
                1                (SBUF(IX),IX=1,(COUNT/DIGITS))
0017    40                      READ(LUNSH,50,END=60)COUNT,
                1                (SHBUF(IX),IX=1,(COUNT/DIGITS))
0018    50                      FORMAT(Q,<BUFLEN>I<DIGITS>)
0019                            IF(COUNT.EQ.0)GO TO 60
0020                            GO TO 70
0021    60                      CONTINUE
0022                                    EOF=YES
0023                                    BUFLEN=(COUNT/DIGITS)
0024                            GO TO 70
0025    70                      CONTINUE
0026                            BPOINT=0
0027                    GO TO 100
0028    80              CONTINUE
                C
                C               CASE:((BPOINT.EQ.BUFLEN).AND.(EOF.EQV.YES))
```

```
C
0029                    IF(.NOT.((BPOINT.EQ.BUFLEN).AND.(EOF.EQV.YES)))GO TO 90
0030                        MORES=NO
0031                    GO TO 100
0032    90              CONTINUE
C
C                       CASE:(SPOINT.EQ.BLKLEN) (DEFAULT)
C
0033                    GO TO 100
0034    100             CONTINUE
0035            GO TO 10
0036    110     CONTINUE
0037            RETURN
0038            END
```

```
DBC.FTN          /TR:BLOCKS/WR

0001             SUBROUTINE DBC(S,SHAT,SPOINT,SIGSTR,SNR,TOTSIG,TOTNOI)
      C
      C          PURPOSE:  TO CALCULATE THE SIGNAL STRENGTH AND SNR FOR A BLOCK
      C                    OF SAMPLES.
      C
      C          V.R      DATE            NAME            COMMENTS
      C          0.1      27-JUN-79       J.M. KRESSE     ORIGINATION
      C          0.2      12-JUL-79       J.M. KRESSE     TO ADD THE SNR HEADER
      C
      C--------DECLARATIONS
      C
0002             PARAMETER MAXVAL=1.E30,MINVAL=-1.E30,REF=1.
0003             INTEGER SPOINT
0004             INTEGER IX,S(SPOINT),SHAT(SPOINT)
0005             REAL NOISE,SIGSQR,SIGSTR,SNR,TOTNOI,TOTSIG
      C
      C--------FUNCTIONS
      C
0006             CLIP(VAL)=AMIN1(AMAX1(VAL,MINVAL),MAXVAL)
0007             DBCLIP(ASIG)=CLIP(10.*ALOG10(ASIG))
      C
      C--------INITIALIZATION
      C
0008             NOISE=0.
0009             SIGSQR=0.
      C
      C--------PROCEDURE
      C
0010             DO 10 IX=1,SPOINT
0011                     SIGSQR=SIGSQR+(FLOAT(S(IX)))**2
0012                     NOISE=NOISE+(FLOAT(S(IX)-SHAT(IX)))**2
0013     10      CONTINUE
0014             TOTSIG=TOTSIG+SIGSQR
0015             TOTNOI=TOTNOI+NOISE
      C
      C          CASE:((SIGSQR.EQ.0.).AND.(NOISE.EQ.0.))
      C
0016             IF(.NOT.((SIGSQR.EQ.0.).AND.(NOISE.EQ.0.)))GO TO 20
0017                     SIGSTR=MINVAL
0018                     SNR=MAXVAL
0019             GO TO 50
0020     20      CONTINUE
      C
      C          CASE:((SIGSQR.NE.0.).AND.(NOISE.EQ.0.))
      C
0021             IF(.NOT.((SIGSQR.NE.0.).AND.(NOISE.EQ.0.)))GO TO 30
0022                     SIGSTR=DBCLIP(SIGSQR/(FLOAT(SPOINT)*REF))
0023                     SNR=MAXVAL
0024             GO TO 50
0025     30      CONTINUE
      C
      C          CASE:((SIGSQR.EQ.0.).AND.(NOISE.NE.0.))
      C
0026             IF(.NOT.((SIGSQR.EQ.0.).AND.(NOISE.NE.0.)))GO TO 40
0027                     SIGSTR=MINVAL
0028                     SNR=MINVAL
```

```
##29                GO TO 50
##30      40        CONTINUE
          C
          C         CASE:((SIGSQR.NE.0.).AND.(NOISE.NE.0.)) (DEFAULT)
          C
##31                        SIGSTR=DBCLIP(SIGSQR/(FLOAT(SPOINT)*REF))
##32                        SNR=DBCLIP(SIGSQR/NOISE)
##33                GO TO 50
##34      50        CONTINUE
##35                RETURN
##36                END
```

[redacted]SEM[redacted] [redacted]TR:[redacted]KS/[redacted]

```
0001            SUBROUTINE PUTSEM(SIGSTR,SNR,LUNSCR)
        C
        C       PURPOSE:  TO WRITE THE SIGNAL STRENGTH AND SNR.
        C
        C       V.R     DATE            NAME            COMMENTS
        C       0.1     27-JUN-79       J.M. KRESSE     ORIGINATION
        C       0.2     12-JUL-79       J.M. KRESSE     TO ADD THE SNR HEADER
        C
        C-------DECLARATIONS
        C
0002            INTEGER LUNSCR
0003            REAL SIGSTR,SNR
        C
        C-------PROCEDURE
        C
0004            WRITE(LUNSCR,10)SIGSTR,SNR
0005    10      FORMAT(2G12.4)
0006            RETURN
0007            END
```

```
0001              SUBROUTINE CLOSER(LUNS,LUNSH,LUNDB,LUNSCR)
                  C
                  C       PURPOSE:  TO CLOSE THE FILES
                  C
                  C       V.R     DATE            NAME            COMMENTS
                  C       0.1     27-JUN-79       J.M. KRESSE     ORIGINATION
                  C       0.2     12-JUL-79       J.M. KRESSE     TO ADD THE SNR HEADER
                  C
                  C-------DECLARATIONS
                  C
0002              INTEGER LUNDB,LUNS,LUNSCR,LUNSH
                  C
                  C-------PROCEDURE
                  C
0003              CLOSE(UNIT=LUNS)
0004              CLOSE(UNIT=LUNSH)
0005              CLOSE(UNIT=LUNDB)
0006              CLOSE(UNIT=LUNSCR)
0007              RETURN
0008              END
```

```
0001            PROGRAM SNRSIG
                C
                C       PURPOSE: TO PLOT LOCAL SNR VS. LOCAL SIGNAL STRENGTH
                C
                C       V.R     NAME            DATE            COMMENTS
                C       0.1     J.M. KRESSE     05-JUL-79       ORIGINATION
                C
                C-------DECLARATIONS
                C
0002            PARAMETER ENDPLT=999,LUNDAT=7,NEWORG=-3,PXMAX=8.5,PXWIND=7.5
0003            PARAMETER PYMAX=10.56,PYWIND=9.5
0004            INTEGER COUNT
0005            REAL PXVO,PYVO,RAWX,RAWXMN,RAWXMX
0006            REAL RAWY,RAWYMN,RAWYMX,SNR,SNRSEG,SUM
0007            REAL VX,VXWIND,VY,VYWIND
                C
                C-------DEFINITIONS
                C
0008            VXWIND=PYWIND
0009            VYWIND=PXWIND
                C
                C-------INITIALIZATION
                C
0010            CALL PLOTS(0,0,0)
0011            COUNT=0
0012            SUM=0.
                C
                C-------PROCEDURE
                C
                C       OPEN THE DATAFILE ON LOGICAL UNIT LUNDAT, AND READ IN THE SNR.
                C
0013            CALL OPEN1(LUNDAT)
0014            READ(LUNDAT,*)SNR
                C
                C       ASK FOR THE PLOTTING RANGES.
                C
0015            CALL RANGE1(RAWXMN,RAWXMX,RAWYMN,RAWYMX)
                C
                C       MOVE THE ORIGIN TO THE VIRTUAL ORIGIN.
                C
0016            PXVO=PXMAX-(PXMAX-PXWIND)/2.
0017            PYVO=(PYMAX-PYWIND)/2.
0018            CALL PLOT(PXVO,PYVO,NEWORG)
                C
                C       DRAW THE VIRTUAL AXES.
                C
0019            CALL AXES1(RAWXMN,RAWXMX,RAWYMN,RAWYMX,VXWIND,VYWIND)
                C
                C       DO UNTIL EOF
                C
0020    10      CONTINUE
0021                    READ(LUNDAT,*,END=20)RAWX,RAWY
0022                    CALL RAWTOV(RAWX,RAWXMN,RAWXMX,VXWIND,VX)
0023                    CALL RAWTOV(RAWY,RAWYMN,RAWYMX,VYWIND,VY)
0024                    COUNT=COUNT+1
0025                    SUM=SUM+VY
```

```
00026                   CALL POINT1(VX,VY)
00027             GO TO 10
00028       20    CONTINUE
            C
            C     COMPUTE SNRSEG.
            C
00029             SUM=SUM/FLOAT(COUNT)
00030             SNRSEG=SUM*(RAWYMX-RAWYMN)/VYWIND+RAWYMN
            C
            C     LABEL THE PLOT.
            C
00031             CALL LABEL1(RAWXMN,RAWXMX,VXWIND,VYWIND,SNR,SNRSEG)
            C
            C     END THE PLOTTING AND CLOSE THE DATAFILE.
            C
00032             CALL PLOT(0.,0.,ENDPLT)
00033             CLOSE(UNIT=LUNDAT)
00034             STOP
00035             END
```

```
0001            SUBROUTINE OPEN1(LUNDAT)
                C
                C       PURPOSE:  TO OPEN THE DATA FILE ON LOGICAL UNIT LUNDAT.
                C
                C       V.R     NAME            DATE            COMMENTS
                C       0.1     J.M. KRESSE     05-JUL-79       ORIGINATION
                C
                C-------DECLARATIONS
                C
0002            PARAMETER DIMENS=33,NUL=0
0003            BYTE DATAFI(DIMENS)
0004            INTEGER LUNDAT
                C
                C-------PROCEDURE
                C
                C       ASK FOR DATAFILENAME.
                C
0005            WRITE(5,10)
0006    10      FORMAT(' WHAT IS THE NAME OF THE DATAFILE TO BE PLOTTED?')
0007            READ(5,20)DATAFI
0008    20      FORMAT(<DIMENS>A1)
                C
                C       INSERT END-OF-STRING MARK.
                C
0009            DATAFI(DIMENS)=NUL
                C
                C       OPEN THE DATAFILE ON LOGICAL UNIT LUNDAT.
                C
0010            OPEN(UNIT=LUNDAT,NAME=DATAFI,TYPE='OLD',SHARED,READONLY)
0011            RETURN
0012            END
```

```
FORTRAN IV-PLUS V02-51          14:39:25     15-APR-80           PAGE 1
RANGE1.FTN      /TR:BLOCKS/WR

0001            SUBROUTINE RANGE1(RAWXMN,RAWXMX,RAWYMN,RAWYMX)
        C
        C       PURPOSE: TO ASK FOR THE PLOTTING RANGES.
        C
        C       V.R     NAME            DATE            COMMENTS
        C       0.1     J.M. KRESSE     09-JUL-79       ORIGINATION
        C
        C-------DECLARATIONS
        C
0002            PARAMETER DEFXMN=-10.,DEFXMX=60.,DEFYMN=-10.,DEFYMX=30.
0003            INTEGER NCHARS
0004            REAL RAWXMN,RAWXMX,RAWYMN,RAWYMX
        C
        C-------PROCEDURE
        C
        C       ASK FOR THE DESIRED RANGE.
        C
0005            WRITE(5,10)DEFXMN,DEFXMX,DEFYMN,DEFYMX
0006    10      FORMAT(' PLEASE ENTER THE RANGES IN THE X AND Y DIRECTIONS,
                1 RESPECTIVELY.'/
                2 ' (FOR THE DEFAULT VALUES OF ',F6.2,', ',F5.2,', ',F6.2,
                3 ', ',F5.2,', HIT RETURN.)')
0007            READ(5,20)NCHARS,RAWXMN,RAWXMX,RAWYMN,RAWYMX
0008    20      FORMAT(Q,4G14.7)
0009            IF(NCHARS.NE.0)GO TO 30
0010                    RAWXMN=DEFXMN
0011                    RAWXMX=DEFXMX
0012                    RAWYMN=DEFYMN
0013                    RAWYMX=DEFYMX
0014            GO TO 30
0015    30      CONTINUE
0016            RETURN
0017            END
```

```
AXES1.FTN       /TR:8LOCKS/WR

0001            SUBROUTINE AXES1(RAWXMN,RAWXMX,RAWYMN,RAWYMX,VXWIND,VYWIND)
      C
      C         PURPOSE: TO DRAW THE AXES.
      C
      C         V.R     NAME            DATE            COMMENTS
      C         0.1     J.M. KRESSE     05-JUL-79       ORIGINATION
      C
      C--------DECLARATIONS
      C
0002            PARAMETER CCW=1,CW=-1,LMASK1=-21846,LMASK2=-30584
0003            PARAMETER MINDIV=5,XUNDIV=10.,YUNDIV=5.
0004            INTEGER DIR,NX,NY,VNX,VNY
0005            REAL RAWXMN,RAWXMX,RAWYMN,RAWYMX,VANGLE,VAXLEN,VDV
0006            REAL VFVAL,VX,VXD,VXFACT,VXWIND,VY,VYD,VYFACT,VYWIND,XD,YD
      C
      C--------PROCEDURE
      C
      C         DRAW THE X AXIS.
      C
0007            VX=0.
0008            CALL RAWTOV(0.,RAWYMN,RAWYMX,VYWIND,VY)
0009            DIR=CW
0010            VAXLEN=VXWIND
0011            VANGLE=0.
0012            VFVAL=RAWXMN
0013            VDV=XUNDIV
0014            VXFACT=VXWIND/((RAWXMX-RAWXMN)/XUNDIV)
0015            CALL VAXIS(VX,VY,DIR,VAXLEN,VANGLE,VFVAL,VDV,VXFACT)
      C
      C         DRAW THE Y AXIS.
      C
0016            CALL RAWTOV(0.,RAWXMN,RAWXMX,VXWIND,VX)
0017            VY=0.
0018            DIR=CCW
0019            VAXLEN=VYWIND
0020            VANGLE=90.
0021            VFVAL=RAWYMN
0022            VDV=YUNDIV
0023            VYFACT=VYWIND/((RAWYMX-RAWYMN)/YUNDIV)
0024            CALL VAXIS(VX,VY,DIR,VAXLEN,VANGLE,VFVAL,VDV,VYFACT)
      C
      C         DRAW THE MAJOR GRID.
      C
0025            VX=0.
0026            VY=0.
0027            VNX=IFIX((RAWXMX-RAWXMN)/XUNDIV)
0028            VXD=VXFACT
0029            VNY=IFIX((RAWYMX-RAWYMN)/YUNDIV)
0030            VYD=VYFACT
0031            CALL ROTATE(VX,VY,VANGLE,X,Y,ANGLE)
0032            CALL ROTATE(VXD,VYD,VANGLE,XD,YD,ANGLE)
0033            NX=VNY
0034            NY=VNX
0035            CALL GRID(X,Y,NX,XD,NY,YD,LMASK1)
      C
      C         DRAW THE MINOR GRID.
```

```
      C
0036            NX=NX*MINDIV
0037            XD=XD/FLOAT(MINDIV)
0038            NY=NY*MINDIV
0039            YD=YD/FLOAT(MINDIV)
0040            CALL GRID(X,Y,NX,XD,NY,YD,LMASK2)
0041            RETURN
0042            END
```

```
0001            SUBROUTINE LABEL1(RAWXMN,RAWXMX,VXWIND,VYWIND,SNR,SNRSEG)
                C
                C       PURPOSE:  TO LABEL THE PLOT.
                C
                C       V.R     DATE            NAME            COMMENTS
                C       0.1     10-JUL-79       J.M. KRESSE     ORIGINATION
                C
0002            PARAMETER HAFLIN=40,HEIGHT=.175,MARGIN=.5
0003            PARAMETER NCHARD=36,NCHR3D=26,VANGLE=0.
0004            INTEGER ASC1(3),ASC2(3)
0005            INTEGER IX,LINE1((NCHARD+1)/2),LINE2(HAFLIN),LINE3((NCHR3D+1)/2)
0006            INTEGER MXCHAR,NCHAR1,NCHAR2,NCHAR3,STR1(2),STR2(5)
0007            REAL ANGLE,RAWXMN,RAWXMX,SNR,SNRSEG,VX,VXWIND,VY,VYWIND,X,Y
0008            EQUIVALENCE (LINE3(1),STR1(1)),(LINE3(3),ASC1(1))
0009            EQUIVALENCE (LINE3(6),STR2(1)),(LINE3(11),ASC2(1))
                C
                C-------DEFINITION
                C
0010            DATA LINE1 /'LO','CA','L ','SN','R ','(D','B)',' V','S.',
                1  ' L','OC','AL',' S','IG','NA','L ','(D','B)'/
0011            DATA STR1 /'SN','R='/
0012            DATA STR2 /'  ',' S','NR','SE','G='/
                C
                C-------FUNCTION
                C
0013            START(NCHAR)=YAXIS+(VXWIND-YAXIS)/2.-(FLOAT(NCHAR)*HEIGHT)/2.
                C
                C-------PROCEDURE
                C
                C       LOCATE THE Y-AXIS.
                C
0014            CALL RAWTOV(0.,RAWXMN,RAWXMX,VXWIND,YAXIS)
                C
                C       CALCULATE THE MAXIMUM NUMBER OF CHARACTERS PERMITTED.
                C
0015            MXCHAR=MIN0(2*HAFLIN,IFIX((VXWIND-YAXIS-2.*MARGIN)/HEIGHT))
0016            NCHAR1=MIN0(MXCHAR,NCHARD)
0017            NCHAR3=MIN0(MXCHAR,NCHR3D)
                C
                C       ASK FOR THE LABELING INFORMATION.
                C
0018    10      CONTINUE
0019                    WRITE(5,20)MXCHAR
0020    20              FORMAT(' HOW DO YOU WANT TO LABEL THE PLOT?'/
                1         ' (',I3,' CHARACTERS, MAXIMUM)')
0021                    READ(5,30)NCHAR2,(LINE2(IX),IX=1,(NCHAR2+1)/2)
0022    30              FORMAT(Q,<(NCHAR2+1)/2>A2)
0023            IF(NCHAR2.GT.MXCHAR)GO TO 10
0024            CONTINUE
                C
                C       START THE FIRST LINE AT THE TOP, CENTERED BETWEEN THE
                C        Y-AXIS AND THE RIGHT EDGE.
                C
0025            VX=START(NCHAR1)
0026            VY=VYWIND-MARGIN-HEIGHT
0027            CALL ROTATE(VX,VY,VANGLE,X,Y,ANGLE)
```

```
0020            CALL SYMBOL(X,Y,HEIGHT,LINE1,ANGLE,NCHAR1)
      C
      C         PLOT THE SECOND LINE.
      C
0029            VX=START(NCHAR2)
0030            VY=VY-2.*HEIGHT
0031            CALL ROTATE(VX,VY,VANGLE,X,Y,ANGLE)
0032            CALL SYMBOL(X,Y,HEIGHT,LINE2,ANGLE,NCHAR2)
      C
      C         GENERATE THE THIRD LINE.
      C
0033            ENCODE(6,40,ASC1)SNR
0034            ENCODE(6,40,ASC2)SNRSEG
0035    40      FORMAT(F6.2)
      C
      C         PLOT THE THIRD LINE.
      C
0036            VX=START(NCHAR3)
0037            VY=VY-2.*HEIGHT
0038            CALL ROTATE(VX,VY,VANGLE,X,Y,ANGLE)
0039            CALL SYMBOL(X,Y,HEIGHT,LINE3,ANGLE,NCHAR3)
0040            RETURN
0041            END
```

VAXIS.FTN /TR:BLOCKS/WR

```
0001            SUBROUTINE VAXIS(VX,VY,DIR,VAXLEN,VANGLE,VFVAL,VDV,FACT)
                C
                C       PURPOSE: TO DRAW A VIRTUAL AXIS.
                C
                C       V.R     NAME            DATE            COMMENTS
                C       0.1     J.M. KRESSE     5-JUL-79        ORIGINATION
                C
                C-------DECLARATIONS
                C
0002            INTEGER DIR
0003            REAL ANGLE,AXLEN,FACT,VANGLE,VAXLEN,VDV,VFVAL,VX,VY,X,Y
                C
                C-------PROCEDURE
                C
                C       ROTATE
                C
0004            CALL ROTATE(VX,VY,VANGLE,X,Y,ANGLE)
                C
                C       FIX ABSOLUTE
                C
0005            X=X/FACT
0006            Y=Y/FACT
0007            AXLEN=VAXLEN/FACT
0008            CALL FACTOR(FACT)
0009            CALL AXIS(X,Y,' ',DIR,AXLEN,ANGLE,VFVAL,VDV)
0010            CALL FACTOR(1.)
0011            RETURN
0012            END
```

```
FORTRAN IV-PLUS V02-51          16:04:17    15-APR-80           PAGE 1
POINT1.FTN      /TR:BLOCKS/WR

0001            SUBROUTINE POINT1(VX,VY)
        C
        C       PURPOSE:  TO PLOT A POINT CORRESPONDING TO (RAWX,RAWY).
        C
        C       V.R     NAME            DATE            COMMENTS
        C       0.1     J.M. KRESSE     10-JUL-79       ORIGINATION
        C
        C-------DECLARATIONS
        C
0002            PARAMETER HEIGHT=.070,ITEXT=11,NC=-1,VANGLE=0.
0003            REAL ANGLE,VX,VY,X,Y
        C
        C-------PROCEDURE
        C
0004            CALL ROTATE(VX,VY,VANGLE,X,Y,ANGLE)
0005            CALL SYMBOL(X,Y,HEIGHT,ITEXT,ANGLE,NC)
0006            RETURN
0007            END
```

```
0001            SUBROUTINE RAWTOV(RAW,RAWMN,RAWMX,VWIND,V)
                C
                C       PURPOSE: TO CONVERT THE RAW DATA TO VIRTUAL DATA.
                C
                C       V.R     DATE            NAME            COMMENTS
                C       0.1     10-JUL-79       J.M. KRESSE     ORIGINATION
                C
                C-------DECLARATIONS
                C
0002            REAL RAW,RAWMN,RAWMX,V,VWIND
                C
                C-------PROCEDURE
                C
0003            V=AMIN1(1.,AMAX1(0.,(RAW-RAWMN)/(RAWMX-RAWMN)))*VWIND
0004            RETURN
0005            END
```

```
ROTATE.FTN      /TR:BLOCKS/WR

0001            SUBROUTINE ROTATE(VX,VY,VANGLE,X,Y,ANGLE)
        C
        C       PURPOSE:  TO ROTATE THE VIRTUAL DATA TO PHYSICAL DATA.
        C
        C       V.R     DATE            NAME            COMMENTS
        C       0.1     10-JUL-79       J.M. KRESSE     ORIGINATION
        C
        C-------DECLARATIONS
        C
0002            REAL ANGLE,VANGLE,VX,VY,X,Y
        C
        C-------PROCEDURE
        C
0003            X=-VY
0004            Y=VX
0005            ANGLE=VANGLE+90.
0006            RETURN
0007            END
```

```
0001            PROGRAM BFCALC
                C
                C       PURPOSE:  TO CALCULATE THE BLOCK SIGNAL STRENGTH AND THE BLOCK
                C                  AVERAGE BUFFER LENGTH.
                C
                C       V.R     DATE            NAME            COMMENTS
                C       0.1     22-JUL-79       J.M. KRESSE     ORIGINATION
                C
                C-------DECLARATIONS
                C
0002            PARAMETER LUNIN=1,LUNOUT=2,LUNTI=5,MAXSIG=60.,MINSIG=-60.
0003            PARAMETER NAMLEN=33,NO=.FALSE.,NUL=0,REF=1,YES=.TRUE.
0004            BYTE FNAME(NAMLEN)
0005            INTEGER BLKLEN,ICNT
0006            LOGICAL EOF
0007            REAL BUFAVG,BUFCNT,BUFSUM,SIG,SIGSTR,SIGSUM
                C
                C-------PROCEDURE
                C
                C       ASK FOR THE INPUT FILENAME, AND OPEN THE INPUTFILE ON LOGICAL
                C        UNIT LUNIN.
                C
0008            WRITE(LUNTI,10)
0009    10      FORMAT(' WHAT IS THE INPUT FILENAME?')
0010            READ(LUNTI,20)FNAME
0011    20      FORMAT(<NAMLEN>A1)
0012            FNAME(NAMLEN)=NUL
0013            OPEN(UNIT=LUNIN,NAME=FNAME,TYPE='OLD',READONLY,SHARED)
                C
                C       ASK FOR THE OUTPUT FILENAME, AND OPEN THE OUTPUTFILE ON LOGICAL
                C        UNIT LUNOUT.
                C
0014            WRITE(LUNTI,30)
0015    30      FORMAT(' WHAT IS THE OUTPUT FILENAME?')
0016            READ(LUNTI,20)FNAME
0017            FNAME(NAMLEN)=NUL
0018            OPEN(UNIT=LUNOUT,NAME=FNAME,TYPE='NEW',CARRIAGECONTROL='LIST')
                C
                C       ASK FOR THE AVERAGING BLOCKLENGTH.
                C
0019            WRITE(LUNTI,40)
0020    40      FORMAT(' WHAT BLOCKLENGTH DO YOU WANT TO USE?')
0021            READ(LUNTI,*)BLKLEN
0022            EOF=NO
                C
                C       DO UNTIL (EOF.EQV.YES)
                C
0023    50      CONTINUE
0024                    SIGSUM=0.
0025                    BUFSUM=0.
0026                    ICNT=0
                C
                C               DO UNTIL ((ICNT.EQ.BLKLEN).OR.(EOF.EQV.YES))
                C
0027    60              CONTINUE
0028                            READ(LUNIN,*,END=70)SIG,BUFCNT
```

```
0029                                    SIGSUM=SIGSUM+ABS(SIG)
0030                                    BUFSUM=BUFSUM+BUFCNT
0031                                    ICNT=ICNT+1
0032                            GO TO 80
0033    70                      CONTINUE
0034                                    EOF=YES
0035                            GO TO 80
0036    80                      CONTINUE
0037                    IF(.NOT.((ICNT.EQ.BLKLEN).OR.(EOF.EQV.YES)))GO TO 60
0038                    CONTINUE
0039                    IF(ICNT.EQ.0)GO TO 120
0040                            IF(SIGSUM.EQ.0.)GO TO 90
0041                                    SIGSTR=AMIN1(MAXSIG,AMAX1(MINSIG,(20.*
     1                                  ALOG10(SIGSUM/(FLOAT(ICNT)*REF)))))
0042                            GO TO 100
0043    90                      CONTINUE
0044                                    SIGSTR=MINSIG
0045                            GO TO 100
0046    100                     CONTINUE
0047                            BUFAVG=BUFSUM/FLOAT(ICNT)
0048                            WRITE(LUNOUT,110)SIGSTR,BUFAVG
0049    110                     FORMAT(F7.2,F10.2)
0050                    GO TO 120
0051    120             CONTINUE
0052            IF(EOF.EQV.NO)GO TO 50
0053            CONTINUE
0054            CLOSE(UNIT=LUNIN)
0055            CLOSE(UNIT=LUNOUT)
0056            STOP
0057            END
```

```
!
!     !     CO    D F    TO    D B    T.T
!
BFPLOT/CP/FP,BFPLOT/NOSP=BFPLOT,OPEN1,RANGE2,LABEL2,AXES2,VAXIS,
POINT1,RAWTOV,ROTATE,[1,54]PEPLIB.OLB/LB
/
UNITS=7
ACTFIL=7
//
```

```
0001            PROGRAM BFPLOT
                C
                C       PURPOSE:  TO PLOT THE AVERAGE BUFFER LENGTH VS. LOCAL SIGNAL
                C                  STRENGTH.
                C
                C       V.R     DATE            NAME            COMMENTS
                C       0.1     23-JUL-79       J.M. KRESSE     ORIGINATION
                C
                C-------DECLARATIONS
                C
0002            PARAMETER ENDPLT=999,LUNDAT=7,NEWORG=-3,PXMAX=8.5,PXWIND=7.5
0003            PARAMETER PYMAX=10.56,PYWIND=9.5
0004            REAL PXVO,PYVO,RAWX,RAWXMN,RAWXMX
0005            REAL RAWY,RAWYMN,RAWYMX
0006            REAL VX,VXWIND,VY,VYWIND
                C
                C-------DEFINITIONS
                C
0007            VXWIND=PYWIND
0008            VYWIND=PXWIND
                C
                C-------INITIALIZATION
                C
0009            CALL PLOTS(0,0,0)
                C
                C-------PROCEDURE
                C
                C       OPEN THE DATAFILE ON LOGICAL UNIT LUNDAT.
                C
0010            CALL OPEN1(LUNDAT)
                C
                C       ASK FOR THE PLOTTING RANGES.
                C
0011            CALL RANGE2(RAWXMN,RAWXMX,RAWYMN,RAWYMX)
                C
                C       MOVE THE ORIGIN TO THE VIRTUAL ORIGIN.
                C
0012            PXVO=PXMAX-(PXMAX-PXWIND)/2.
0013            PYVO=(PYMAX-PYWIND)/2.
0014            CALL PLOT(PXVO,PYVO,NEWORG)
                C
                C       DRAW THE VIRTUAL AXES.
                C
0015            CALL AXES2(RAWXMN,RAWXMX,RAWYMN,RAWYMX,VXWIND,VYWIND)
                C
                C       DO UNTIL EOF
                C
0016    10      CONTINUE
0017                    READ(LUNDAT,*,END=20)RAWX,RAWY
0018                    CALL RAWTOV(RAWX,RAWXMN,RAWXMX,VXWIND,VX)
0019                    CALL RAWTOV(RAWY,RAWYMN,RAWYMX,VYWIND,VY)
0020                    CALL POINT1(VX,VY)
0021            GO TO 10
0022    20      CONTINUE
                C
                C       LABEL THE PLOT.
```

```
C
      CALL LABEL2(RAWXMN,RAWXMX,VXWIND,VYWIND)
C
C     END THE PLOTTING AND CLOSE THE DATAFILE.
C
      CALL PLOT(0.,0.,ENDPLT)
      CLOSE(UNIT=LUNDAT)
      STOP
      END
```

```
FORTRAN IV-PLUS V02-51          14:41:43    15-APR-80           PAGE 1
RANGE2.FTN      /TR:BLOCKS/WR

0001            SUBROUTINE RANGE2(RAWXMN,RAWXMX,RAWYMN,RAWYMX)
        C
        C       PURPOSE: TO ASK FOR THE PLOTTING RANGES.
        C
        C       V.R     DATE            NAME            COMMENTS
        C       0.1     23-JUL-79       J.M. KRESSE     ORIGINATION
        C
        C-------DECLARATIONS
        C
0002            PARAMETER DEFXMN=-10.,DEFXMX=60.,DEFYMN=-2000.,DEFYMX=500.
0003            INTEGER NCHARS
0004            REAL RAWXMN,RAWXMX,RAWYMN,RAWYMX
        C
        C-------PROCEDURE
        C
        C       ASK FOR THE DESIRED RANGE.
        C
0005            WRITE(5,10)DEFXMN,DEFXMX,DEFYMN,DEFYMX
0006    10      FORMAT(' PLEASE ENTER THE RANGES IN THE X AND Y DIRECTIONS,
                1 RESPECTIVELY.'/
                2 ' (FOR THE DEFAULT VALUES OF ',F6.2,', ',F5.2,', ',F8.2,
                3 ', ',F6.2,', HIT RETURN.)')
0007            READ(5,20)NCHARS,RAWXMN,RAWXMX,RAWYMN,RAWYMX
0008    20      FORMAT(Q,4G14.7)
0009            IF(NCHARS.NE.0)GO TO 30
0010                    RAWXMN=DEFXMN
0011                    RAWXMX=DEFXMX
0012                    RAWYMN=DEFYMN
0013                    RAWYMX=DEFYMX
0014            GO TO 30
0015    30      CONTINUE
0016            RETURN
0017            END
```

```
0001            SUBROUTINE AXES2(RAWXMN,RAWXMX,RAWYMN,RAWYMX,VXWIND,VYWIND)
                C
                C       PURPOSE: TO DRAW THE AXES.
                C
                C       V.R     DATE            NAME            COMMENTS
                C       0.1     23-JUL-79       J.M. KRESSE     ORIGINATION
                C
                C-------DECLARATIONS
                C
0002            PARAMETER CCW=1,CW=-1,LMASK1=-21846,LMASK2=-30584
0003            PARAMETER MINDIV=5,XUNDIV=10.,YUNDIV=500.
0004            INTEGER DIR,NX,NY,VNX,VNY
0005            REAL RAWXMN,RAWXMX,RAWYMN,RAWYMX,VANGLE,VAXLEN,VDV
0006            REAL VFVAL,VX,VXD,VXFACT,VXWIND,VY,VYD,VYFACT,VYWIND,XD,YD
                C
                C-------PROCEDURE
                C
                C       DRAW THE X AXIS.
                C
0007            VX=0.
0008            CALL RAWTOV(0.,RAWYMN,RAWYMX,VYWIND,VY)
0009            DIR=CW
0010            VAXLEN=VXWIND
0011            VANGLE=0.
0012            VFVAL=RAWXMN
0013            VDV=XUNDIV
0014            VXFACT=VXWIND/((RAWXMX-RAWXMN)/XUNDIV)
0015            CALL VAXIS(VX,VY,DIR,VAXLEN,VANGLE,VFVAL,VDV,VXFACT)
                C
                C       DRAW THE Y AXIS.
                C
0016            CALL RAWTOV(0.,RAWXMN,RAWXMX,VXWIND,VX)
0017            VY=0.
0018            DIR=CCW
0019            VAXLEN=VYWIND
0020            VANGLE=90.
0021            VFVAL=RAWYMN
0022            VDV=YUNDIV
0023            VYFACT=VYWIND/((RAWYMX-RAWYMN)/YUNDIV)
0024            CALL VAXIS(VX,VY,DIR,VAXLEN,VANGLE,VFVAL,VDV,VYFACT)
                C
                C       DRAW THE MAJOR GRID.
                C
0025            VX=0.
0026            VY=0.
0027            VNX=IFIX((RAWXMX-RAWXMN)/XUNDIV)
0028            VXD=VXFACT
0029            VNY=IFIX((RAWYMX-RAWYMN)/YUNDIV)
0030            VYD=VYFACT
0031            CALL ROTATE(VX,VY,VANGLE,X,Y,ANGLE)
0032            CALL ROTATE(VXD,VYD,VANGLE,XD,YD,ANGLE)
0033            NX=VNY
0034            NY=VNX
0035            CALL GRID(X,Y,NX,XD,NY,YD,LMASK1)
                C
                C       DRAW THE MINOR GRID.
```

```
      C
##36            NX=NX*MINDIV
##37            XD=XD/FLOAT(MINDIV)
##38            NY=NY*MINDIV
##39            YD=YD/FLOAT(MINDIV)
##40            CALL GRID(X,Y,NX,XD,NY,YD,LMASK2)
##41            RETURN
##42            END
```

```
0001            SUBROUTINE LABEL2(RAWXMN,RAWXMX,VXWIND,VYWIND)
        C
        C       PURPOSE:  TO LABEL THE PLOT.
        C
        C       V.R     DATE            NAME            COMMENTS
        C       00.1    23-JUL-79       J.M. KRESSE     ORIGINATION
        C
0002            PARAMETER HAFLIN=40,HEIGHT=.175,MARGIN=.5
0003            PARAMETER NCHARD=36,VANGLE=0.
0004            INTEGER IX,LINE1((NCHARD+1)/2),LINE2(HAFLIN)
0005            INTEGER MXCHAR,NCHAR1,NCHAR2
0006            REAL ANGLE,RAWXMN,RAWXMX,VX,VXWIND,VY,VYWIND,X,Y
        C
        C-------DEFINITION
        C
0007            DATA LINE1 /'AV','G ','BU','F ','LE','NG','TH',' V','S.',
                1  ' L','OC','AL',' S','IG','NA','L ','(D','B)'/
        C
        C-------FUNCTION
        C
0008            START(NCHAR)=YAXIS+(VXWIND-YAXIS)/2.-(FLOAT(NCHAR)*HEIGHT)/2.
        C
        C-------PROCEDURE
        C
        C       LOCATE THE Y-AXIS.
        C
0009            CALL RAWTOV(0.,RAWXMN,RAWXMX,VXWIND,YAXIS)
        C
        C       CALCULATE THE MAXIMUM NUMBER OF CHARACTERS PERMITTED.
        C
0010            MXCHAR=MIN0(2*HAFLIN,IFIX((VXWIND-YAXIS-2.*MARGIN)/HEIGHT))
0011            NCHAR1=MIN0(MXCHAR,NCHARD)
        C
        C       ASK FOR THE LABELING INFORMATION.
        C
0012    10      CONTINUE
0013                    WRITE(5,20)MXCHAR
0014    20              FORMAT(' HOW DO YOU WANT TO LABEL THE PLOT?'/
                1        ' (',I3,' CHARACTERS, MAXIMUM)')
0015                    READ(5,30)NCHAR2,(LINE2(IX),IX=1,(NCHAR2+1)/2)
0016    30              FORMAT(Q,<(NCHAR2+1)/2>A2)
0017            IF(NCHAR2.GT.MXCHAR)GO TO 10
0018            CONTINUE
        C
        C       START THE FIRST LINE AT THE TOP, CENTERED BETWEEN THE
        C        Y-AXIS AND THE RIGHT EDGE.
        C
0019            VX=START(NCHAR1)
0020            VY=VYWIND-MARGIN-HEIGHT
0021            CALL ROTATE(VX,VY,VANGLE,X,Y,ANGLE)
0022            CALL SYMBOL(X,Y,HEIGHT,LINE1,ANGLE,NCHAR1)
        C
        C       PLOT THE SECOND LINE.
        C
0023            VX=START(NCHAR2)
0024            VY=VY-2.*HEIGHT
```

```
0025            CALL ROTATE(VX,VY,VANGLE,X,Y,ANGLE)
0026            CALL SYMBOL(X,Y,HEIGHT,LINE2,ANGLE,NCHAR2)
0027            RETURN
0028            END
```

```
0001            PROGRAM BFDIFF
                C
                C       PURPOSE:  TO CALCULATE THE BLOCK SIGNAL STRENGTH AND THE
                C                  DIFFERENCE IN THE AVERAGE BUFFER LENGTH.
                C
                C       V.R     DATE            NAME            COMMENTS
                C       0.1     23-JUL-79       J.M. KRESSE     ORIGINATION
                C
                C-------DECLARATIONS
                C
0002            PARAMETER LUNIN=1,LUNOUT=2,LUNTI=5
0003            PARAMETER NAMLEN=33,NUL=0
0004            BYTE FNAME(NAMLEN)
0005            REAL BUFAVG,LAST,SIGSTR
                C
                C-------PROCEDURE
                C
                C       ASK FOR THE INPUT FILENAME, AND OPEN THE INPUTFILE ON LOGICAL
                C        UNIT LUNIN.
                C
0006            WRITE(LUNTI,10)
0007    10      FORMAT(' WHAT IS THE INPUT FILENAME?')
0008            READ(LUNTI,20)FNAME
0009    20      FORMAT(<NAMLEN>A1)
0010            FNAME(NAMLEN)=NUL
0011            OPEN(UNIT=LUNIN,NAME=FNAME,TYPE='OLD',READONLY,SHARED)
                C
                C       ASK FOR THE OUTPUT FILENAME, AND OPEN THE OUTPUTFILE ON LOGICAL
                C        UNIT LUNOUT.
                C
0012            WRITE(LUNTI,30)
0013    30      FORMAT(' WHAT IS THE OUTPUT FILENAME?')
0014            READ(LUNTI,20)FNAME
0015            FNAME(NAMLEN)=NUL
0016            OPEN(UNIT=LUNOUT,NAME=FNAME,TYPE='NEW',CARRIAGECONTROL='LIST')
0017            LAST=0.
                C
                C       DO UNTIL EOF
                C
0018    50      CONTINUE
0019                    READ(LUNIN,*,END=70)SIGSTR,BUFAVG
0020                    BUFDIF=BUFAVG-LAST
0021                    WRITE(LUNOUT,60)SIGSTR,BUFDIF
0022    60              FORMAT(F7.2,F10.2)
0023                    LAST=BUFAVG
0024            GO TO 50
0025    70      CONTINUE
0026            CLOSE(UNIT=LUNIN)
0027            CLOSE(UNIT=LUNOUT)
0028            STOP
0029            END
```

```
!
!          COMMAND FILE TO BUILD BFDFPL.TSK
!
BFDFPL/CP/FP,BFDFPL/NOSP=BFDFPL,OPEN1,RANGE3,LABEL3,AXES3,VAXIS,
POINT1,RAWTOV,ROTATE,[1,54]PEPLIB.OLB/LB
/
UNITS=7
ACTFIL=7
//
```

```
0001                PROGRAM BFOFPL
                C
                C       PURPOSE:  TO PLOT THE DIFFERENCE IN THE AVERAGE BUFFER LENGTH
                C                  VS. THE LOCAL SIGNAL STRENGTH.
                C
                C       V.R     DATE            NAME            COMMENTS
                C       0.1     23-JUL-79       J.M. KRESSE     ORIGINATION
                C
                C-------DECLARATIONS
                C
0002                PARAMETER ENDPLT=999,LUNDAT=7,NEWORG=-3,PXMAX=8.5,PXWIND=7.5
0003                PARAMETER PYMAX=10.56,PYWIND=9.5
0004                REAL PXVO,PYVO,RAWX,RAWXMN,RAWXMX
0005                REAL RAWY,RAWYMN,RAWYMX
0006                REAL VX,VXWIND,VY,VYWIND
                C
                C-------DEFINITIONS
                C
0007                VXWIND=PYWIND
0008                VYWIND=PXWIND
                C
                C-------INITIALIZATION
                C
0009                CALL PLOTS(0,0,0)
                C
                C-------PROCEDURE
                C
                C       OPEN THE DATAFILE ON LOGICAL UNIT LUNDAT.
                C
0010                CALL OPEN1(LUNDAT)
                C
                C       ASK FOR THE PLOTTING RANGES.
                C
0011                CALL RANGE3(RAWXMN,RAWXMX,RAWYMN,RAWYMX)
                C
                C       MOVE THE ORIGIN TO THE VIRTUAL ORIGIN.
                C
0012                PXVO=PXMAX-(PXMAX-PXWIND)/2.
0013                PYVO=(PYMAX-PYWIND)/2.
0014                CALL PLOT(PXVO,PYVO,NEWORG)
                C
                C       DRAW THE VIRTUAL AXES.
                C
0015                CALL AXES3(RAWXMN,RAWXMX,RAWYMN,RAWYMX,VXWIND,VYWIND)
                C
                C       DO UNTIL EOF
                C
0016    10      CONTINUE
0017                    READ(LUNDAT,*,END=20)RAWX,RAWY
0018                    CALL RAWTOV(RAWX,RAWXMN,RAWXMX,VXWIND,VX)
0019                    CALL RAWTOV(RAWY,RAWYMN,RAWYMX,VYWIND,VY)
0020                    CALL POINT1(VX,VY)
0021                GO TO 10
0022    20      CONTINUE
                C
                C       LABEL THE PLOT.
```

```
C
      CALL LABEL3(RAWXMN,RAWXMX,VXWIND,VYWIND)
C
C     END THE PLOTTING AND CLOSE THE DATAFILE.
C
      CALL PLOT(0.,0.,ENDPLT)
      CLOSE(UNIT=LUNDAT)
      STOP
      END
```

RANGE3.FTN /TR:BLOCKS/WR

```
0001            SUBROUTINE RANGE3(RAWXMN,RAWXMX,RAWYMN,RAWYMX)
                C
                C       PURPOSE: TO ASK FOR THE PLOTTING RANGES.
                C
                C       V.R     DATE            NAME            COMMENTS
                C       0.1     23-JUL-79       J.M. KRESSE     ORIGINATION
                C
                C-------DECLARATIONS
                C
0002            PARAMETER DEFXMN=-10.,DEFXMX=60.,DEFYMN=-130.,DEFYMX=120.
0003            INTEGER NCHARS
0004            REAL RAWXMN,RAWXMX,RAWYMN,RAWYMX
                C
                C-------PROCEDURE
                C
                C       ASK FOR THE DESIRED RANGE.
                C
0005            WRITE(5,10)DEFXMN,DEFXMX,DEFYMN,DEFYMX
0006    10      FORMAT(' PLEASE ENTER THE RANGES IN THE X AND Y DIRECTIONS,
                1 RESPECTIVELY.'/
                2 ' (FOR THE DEFAULT VALUES OF ',F6.2,', ',F5.2,', ',F8.2,
                3 ', ',F6.2,', HIT RETURN.)')
0007            READ(5,20)NCHARS,RAWXMN,RAWXMX,RAWYMN,RAWYMX
0008    20      FORMAT(Q,4G14.7)
0009            IF(NCHARS.NE.0)GO TO 30
0010                    RAWXMN=DEFXMN
0011                    RAWXMX=DEFXMX
0012                    RAWYMN=DEFYMN
0013                    RAWYMX=DEFYMX
0014            GO TO 30
0015    30      CONTINUE
0016            RETURN
0017            END
```

```
0001            SUBROUTINE AXES3(RAWXMN,RAWXMX,RAWYMN,RAWYMX,VXWIND,VYWIND)
                C
                C       PURPOSE: TO DRAW THE AXES.
                C
                C       V.R     DATE            NAME            COMMENTS
                C       0.1     23-JUL-79       J.M. KRESSE     ORIGINATION
                C
                C-------DECLARATIONS
                C
0002            PARAMETER CCW=1,CW=-1,LMASK1=-21846,LMASK2=-30584
0003            PARAMETER MINDIV=5,XUNDIV=10.,YUNDIV=50.
0004            INTEGER DIR,NX,NY,VNX,VNY
0005            REAL RAWXMN,RAWXMX,RAWYMN,RAWYMX,VANGLE,VAXLEN,VDV
0006            REAL VFVAL,VX,VXD,VXFACT,VXWIND,VY,VYD,VYFACT,VYWIND,XD,YD
                C
                C-------PROCEDURE
                C
                C       DRAW THE X AXIS.
                C
0007            VX=0.
0008            CALL RAWTOV(0.,RAWYMN,RAWYMX,VYWIND,VY)
0009            DIR=CW
0010            VAXLEN=VXWIND
0011            VANGLE=0.
0012            VFVAL=RAWXMN
0013            VDV=XUNDIV
0014            VXFACT=VXWIND/((RAWXMX-RAWXMN)/XUNDIV)
0015            CALL VAXIS(VX,VY,DIR,VAXLEN,VANGLE,VFVAL,VDV,VXFACT)
                C
                C       DRAW THE Y AXIS.
                C
0016            CALL RAWTOV(0.,RAWXMN,RAWXMX,VXWIND,VX)
0017            VY=0.
0018            DIR=CCW
0019            VAXLEN=VYWIND
0020            VANGLE=90.
0021            VFVAL=RAWYMN
0022            VDV=YUNDIV
0023            VYFACT=VYWIND/((RAWYMX-RAWYMN)/YUNDIV)
0024            CALL VAXIS(VX,VY,DIR,VAXLEN,VANGLE,VFVAL,VDV,VYFACT)
                C
                C       DRAW THE MAJOR GRID.
                C
0025            VX=0.
0026            VY=0.
0027            VNX=IFIX((RAWXMX-RAWXMN)/XUNDIV)
0028            VXD=VXFACT
0029            VNY=IFIX((RAWYMX-RAWYMN)/YUNDIV)
0030            VYD=VYFACT
0031            CALL ROTATE(VX,VY,VANGLE,X,Y,ANGLE)
0032            CALL ROTATE(VXD,VYD,VANGLE,XD,YD,ANGLE)
0033            NX=VNY
0034            NY=VNX
0035            CALL GRID(X,Y,NX,XD,NY,YD,LMASK1)
                C
                C       DRAW THE MINOR GRID.
```

```
C
      NX=NX*MINDIV
      XD=XD/FLOAT(MINDIV)
      NY=NY*MINDIV
      YD=YD/FLOAT(MINDIV)
      CALL GRID(X,Y,NX,XD,NY,YD,LMASK2)
      RETURN
      END
```

```
FORTRAN IV-PLUS V02-51          14:43:38     15-APR-80          PAGE 1
LABEL3.FTN      /TR:BLOCKS/WR

0001            SUBROUTINE LABEL3(RAWXMN,RAWXMX,VXWIND,VYWIND)
        C
        C       PURPOSE:  TO LABEL THE PLOT.
        C
        C       V.R     DATE            NAME            COMMENTS
        C       0.1     23-JUL-79       J.M. KRESSE     ORIGINATION
        C
0002            PARAMETER HAFLIN=40,HEIGHT=.175,MARGIN=.5
0003            PARAMETER NCHARD=36,VANGLE=0.
0004            INTEGER IX,LINE1((NCHARD+1)/2),LINE2(HAFLIN)
0005            INTEGER MXCHAR,NCHAR1,NCHAR2
0006            REAL ANGLE,RAWXMN,RAWXMX,VX,VXWIND,VY,VYWIND,X,Y
        C
        C-------DEFINITION
        C
0007            DATA LINE1 /'DI','F ','AV','G ','BU','FF','ER',' V','S.',
                1 ' L','OC','AL',' S','IG','NA','L ','(D','B)'/
        C
        C-------FUNCTION
        C
0008            START(NCHAR)=YAXIS+(VXWIND-YAXIS)/2.-(FLOAT(NCHAR)*HEIGHT)/2.
        C
        C-------PROCEDURE
        C
        C       LOCATE THE Y-AXIS.
        C
0009            CALL RAWTOV(0.,RAWXMN,RAWXMX,VXWIND,YAXIS)
        C
        C       CALCULATE THE MAXIMUM NUMBER OF CHARACTERS PERMITTED.
        C
0010            MXCHAR=MIN0(2*HAFLIN,IFIX((VXWIND-YAXIS-2.*MARGIN)/HEIGHT))
0011            NCHAR1=MIN0(MXCHAR,NCHARD)
        C
        C       ASK FOR THE LABELING INFORMATION.
        C
0012    10      CONTINUE
0013                    WRITE(5,20)MXCHAR
0014    20              FORMAT(' HOW DO YOU WANT TO LABEL THE PLOT?'/
                1         ' (',I3,' CHARACTERS, MAXIMUM)')
0015                    READ(5,30)NCHAR2,(LINE2(IX),IX=1,(NCHAR2+1)/2)
0016    30              FORMAT(Q,<(NCHAR2+1)/2>A2)
0017            IF(NCHAR2.GT.MXCHAR)GO TO 10
0018            CONTINUE
        C
        C       START THE FIRST LINE AT THE TOP, CENTERED BETWEEN THE
        C        Y-AXIS AND THE RIGHT EDGE.
        C
0019            VX=START(NCHAR1)
0020            VY=VYWIND-MARGIN-HEIGHT
0021            CALL ROTATE(VX,VY,VANGLE,X,Y,ANGLE)
0022            CALL SYMBOL(X,Y,HEIGHT,LINE1,ANGLE,NCHAR1)
        C
        C       PLOT THE SECOND LINE.
        C
0023            VX=START(NCHAR2)
0024            VY=VY-2.*HEIGHT
```

```
LABEL3.FTN      /TR:BLOCKS/WR

0025            CALL ROTATE(VX,VY,VANGLE,X,Y,ANGLE)
0026            CALL SYMBOL(X,Y,HEIGHT,LINE2,ANGLE,NCHAR2)
0027            RETURN
0028            END
```

APPENDIX C

PLOTTING PROGRAM

This program is used to plot a data file or the difference of two data file on a Versaplot 07 system. It employs the Versaplot-07 PPEP Software Package.

Two options are provided by this program:

1. It can plot a whole data file. The ranges of x-axis and y-axis are specified by the user through a terminal.

2. It can plot a number of sections of a data file. The starting location and number of sections can be specified by the user through a terminal. However, the size of a section is fixed to 1600 samples.

Data must be stored in 1615 format with a standard format header card. Details are shown in the program listing.

```
C
C       ************************************************
C       *                                              *
C       *       VERSATEC DATAFILE PLOTING PROGRAM      *
C       *                                              *
C       ************************************************
C
C       MAWLIN YEH
C       DATE:6/28/79
C       PROGRAM NAME:   DATAPLOT.FTN
C
C       THIS PROGRAM IS USED TO PLOT A SPEECH SAMPLE FILE ON
C       ON A VERSATEC PRINTER/PLOTTER. THE FORMAT FORMAT FOR
C       THE SPEECH SAMPLES IS 16I5.
C
C       THERE ARE TWO FUNCTIONS IN THIS PROGRAM:
C       1. PLOT A DATA FILE
C       2. PLOT THE DIFFERENCE BETWEEN TWO DATA FILES
C
C       ALSO, TWO OPTIONS ARE AVAILABLE AS FOLLLOWS:
C       1. PLOT THE WHOLE DATA FILE.
C       2. PLOT PART OF DATA FILE.
C
0001            DIMENSION X(1602),Y(1602)
0002            LOGICAL*1 NAME1(32),NAME2(32),SYM1(80),SYM2(80)
0003            INTEGER*2 H1(40),H2(40),X1(16),X2(16),X3(16)
C
C       ASK FOR THE CHOICE
C
0004    10      TYPE *,' WHICH FUNCTION DO YOU WANT?'
0005            TYPE *,' 1 = PLOT A DATA FILE'
0006            TYPE *,' 2 = PLOT THE DIFFERENCE BETWEEN TWO FILE (FILE1-FILE2)'
0007            ACCEPT *,IOPT
0008            TYPE *,' DO YOU WANT TO PLOT THE WHOLE DATA FILE?(1=YES,2=NO)'
0009            ACCEPT *,IOPT2
0010            IF(IOPT2 .EQ. 1)GO TO 70
0011            TYPE *,' WHAT IS THE STARTING SAMPLE NUMBER?'
0012            ACCEPT *,ISTART
0013            TYPE *,' NUMBER OF WINDOWS YOU WANT TO PLOT?'
0014            TYPE *,' ( WINDOW SIZE = 1600 SAMPLES )'
0015            ACCEPT *,NWINDO
0016    70      IF(IOPT .EQ. 1) GO TO 20
0017            IF(IOPT .NE. 2) GO TO 10
C
C       FUNCTION 2
C
0018            TYPE *,' INPUT 1ST FILE NAME?'
0019            ACCEPT 1000,NAME1
0020            NAME1(32)=0
0021            OPEN(UNIT=9,NAME=NAME1,READONLY,SHARED,TYPE='OLD')
0022            READ(9,1001)NSENT1,IRATE1,NSAMP1,IUPER1,ILOWR1,NTERM1,H1
0023            TYPE *,' 1ST FILE HEADER IS'
0024            TYPE 1001,NSENT1,IRATE1,NSAMP1,IUPER1,ILOWR1,NTERM1,H1
0025            TYPE *,' THE LABEL OF THIS FILE FOR THE PLOT?'
0026            TYPE *,' ( 80 CHARACTERS, MAXIMUM )'
0027            ACCEPT 1006,SYM1
```

```
FORTRAN IV-PLUS V02-51          11:17:25    31-MAR-80          PAGE 2
DATAPLOT.FTN    /TR:BLOCKS/WR

0028            TYPE *,' INPUT 2ND FILE NAME?'
0029            ACCEPT 1000,NAME2
0030            NAME2(32)=0
0031            OPEN(UNIT=7,NAME=NAME2,READONLY,SHARED,TYPE='OLD')
0032            READ(7,1001)NSENT2,IRATE2,NSAMP2,IUPER2,ILOWR2,NTERM2,H2
0033            TYPE *,' 2ND FILE HEADER IS'
0034            TYPE 1001,NSENT2,IRATE2,NSAMP2,IUPER2,ILOWR2,NTERM2,H2
0035            TYPE *,' THE LABEL OF THIS FILE FOR THE PLOT?'
0036            TYPE *,' ( 80 CHARACTERS, MAXIMUM )'
0037            ACCEPT 1006,SYM2
      C
      C         PRODUCE A SCRATCH FILE
      C
0038            OPEN(UNIT=8,NAME='SCRATCH.DAT',TYPE='SCRATCH')
0039      50    READ(9,1004,END=40)X1
0040            READ(7,1004,END=40)X2
0041            DO 30 I=1,16
0042      30    X3(I)=X1(I)-X2(I)
0043            WRITE(8,1004)X3
0044            GO TO 50
0045      40    REWIND 8
0046            GO TO 60
0047      20    TYPE *,' INPUT FILE NAME?'
0048            ACCEPT 1000,NAME1
0049            NAME1(32)=0
0050            OPEN(UNIT=8,NAME=NAME1,READONLY,SHARED,TYPE='OLD')
0051            TYPE *,' INPUT FILE HEADER IS'
0052            READ(8,1001)NSENT,IRATE,NSAMP,IUPER,ILOWR,NTERMS,H1
0053            TYPE 1001,NSENT,IRATE,NSAMP,IUPER,ILOWR,NTERMS,H1
0054            TYPE *,' THE LABEL OF THIS FILE FOR THE PLOT?'
0055            TYPE *,' ( 80 CHARACTERS, MAXIMUM )'
0056            ACCEPT 1006,SYM1
0057      1001  FORMAT(6I5,10X,40A1)
0058      1000  FORMAT(32A1)
0059      1004  FORMAT(16I5)
0060      1006  FORMAT(80A1)
      C
      C         PLOTTING ROUTINE
      C
0061      60    TYPE *,' THE WIDTH FOR X-AXIS IS "X-WIDTH" INCHES FOR 1600 SAMPLES'
0062            TYPE *,' THE WIDTH FOR Y-AXIS IS 8 INCHES.'
0063            TYPE *,' INPUT WINDOW SIZE?(XMAX,XMIN,YMAX,YMIN)'
0064            TYPE *,' E.G. 144. 0. 10. 0.'
0065            ACCEPT *,XMAX,XMIN,YMAX,YMIN
0066            TYPE *,' X-WIDTH,X-SCALING FACTOR,Y-STARTING VALUE,Y-SCALING FACTOR ?'
0067            TYPE *,' E.G. 16. 100. -2400. 600. ( FOR ONE FILE )'
0068            TYPE *,' E.G. 16. 100. -600. 150. ( FOR THE DIFFERENCE FILE )'
0069            ACCEPT *,XWIDTH,XSCAL,YSTART,YSCAL
0070            IF(IOPT2 .EQ. 1)GO TO 80
0071            ISAMP=16
0072      90    IF(ISAMP .GE. ISTART)GO TO 100
0073            ISAMP=ISAMP+16
0074            READ(8,1004)X1
0075            GO TO 90
0076      100   SAMP=ISAMP
0077            SAMP=ISAMP
```

```
0078              FA=SAMP
0079     80       DO 100 I=1,1600,16
0080              READ(8,1004)X1
0081              DO 200 J=1,16
0082     200      Y(I+J-1)=X1(J)
0083     100      CONTINUE
0084              DO 300 I=1,1500
0085              FA=FA+1.
0086     300      X(I)=FA
0087              CALL PLOTS(0.,0.,0.)
0088              CALL PLOT(0.,0.,-3)
0089              CALL WINDOW(XMAX,XMIN,YMAX,YMIN)
0090              CALL SYMBOL(2.,9.,0.14,SYM1,0.,80)
0091              IF(IOPT .EQ. 1) GO TO 555
0092              CALL SYMBOL(2.,8.5,0.14,SYM2,0.,80)
0093     555      CALL PLOT(1.,1.,-3)
0094              X(1601)=SAMP
0095              X(1602)=XSCAL
0096              Y(1601)=YSTART
0097              Y(1602)=YSCAL
0098              CALL AXIS(0.,0.,13HSAMPLE NUMBER,-13,XWIDTH,0.,
         1   X(1601),X(1602))
0099              CALL AXIS(0.,0.,9HAMPLITUDE,9,8.,90.,
         1   Y(1601),Y(1602))
0100     500      CALL LINE(X,Y,1600,1,0,0)
0101              IF(IOPT2 .EQ. 1) GO TO 501
0102              NWINDO=NWINDO-1
0103              IF(NWINDO .EQ. 0) GO TO 110
0104     501      XTEMP=X(1600)
0105              YTEMP=Y(1600)
0106              ICOUNT=0.
0107     600      READ(8,1004,END=700)X1
0108              DO 333 J=1,16
0109              ICOUNT=ICOUNT+1
0110     333      Y(ICOUNT)=X1(J)
0111              IF(ICOUNT .NE. 1600) GO TO 600
0112              GO TO 800
0113     700      DO 900 I=ICOUNT+1,1600
0114              IEND=1
0115     900      Y(I)=0.
0116     800      DO 444 I=1,1500
0117              FA=FA+1.
0118     444      X(I)=FA
0119              XX=(X(1)-X(1601))/X(1602)
0120              YY=(Y(1)-Y(1601))/Y(1602)
0121              XTEMP1=(XTEMP-X(1601))/X(1602)
0122              YTEMP1=(YTEMP-Y(1601))/Y(1602)
0123              CALL PLOT(XTEMP1,YTEMP1,3)
0124              CALL PLOT(XX,YY,2)
0125              X(1601)=X(1601)+1600.
0126              CALL PLOT(16.,0.,-3)
0127              CALL AXIS(0.,0.,13HSAMPLE NUMBER,-13,16.,0.,
         1   X(1601),X(1602))
0128              IF(IEND .NE. 1) GO TO 500
0129              CALL LINE(X,Y,1600,1,0,0)
0130     110      CALL PLOT(0.,0.,999)
```

```
FORTRAN IV-PLUS V02-51          11:17:25    31-MAR-80          PAGE 4
DATAPLOT.FTN    /TR:BLOCKS/WR

#131            CALL CLOSE(8)
#132            STOP
#133            END
```

APPENDIX D

PDP-11 D/A PROGRAMS

This set of D/A programs is used for a digital-to-analog converting operation employing the DATEL's ST-PDP device which is an analog I/O module for DEC PDP-11 minicomputers. The ST-PDP device has three different modes:

1. Program Control Interface
2. Interrupt Serviced Interface
3. Direct Memory Address

The set of D/A programs uses the Program Control Interface mode.

The set consists of six modules:

1. DA.CMD —— The indirect command file to execute the D/A converting operation.
2. DASP.FTN —— The Fortran file to output a speech data file to the D/A converter.
3. DARAMP.FTN —— The Fortran file to output a ramp function to the D/A converter in order to check the timing and operation.
4. DA64.MAC —— The MACRO-11 assembly language subroutine for 6400 samples per second sampling frequency.
5. DA80.MAC —— The MACRO-11 assembly language subroutine for 8000 samples per second sampling frequency.
6. COMMON.MAC —— The MACRO-11 assembly language file used to build a common device block inside the PDP-11 operating system.

This set of programs allows a user to output a data file from disk to the D/A converter. Data must be stored in 1615 format with a standard format header card. Details are shown in the program listing.

A common device block has to be built the first time the D/A device is used. The way to build is shown below. Further details can be found in the I/O Driver Reference Manual of the PDP-11 RMS-11M operating system.

1. Logon a privileged UIC: For future reference, use UIC = [3,1]
2. > MAC COMMON = COMMON
   (underline means the prompt of the computer)
3. > SET /UIC = [1,1]
4. > TKB
   TKB> COMMON/MM, LP:, ST: COMMON/PI/-HD = [3,1] COMMON
   TKB> /
   ENTER OPTIONS:
   TKB> PAR = COMMON: 0:16000
   TKB> STACK = 0
   TKB> 1
5. Logoff

The procedure to execute the D/A modules is as follows:

1. Logon a privileged UIC
2. Execute the indirect command file DA.CMD (i.e., TYPE @ DA)

During the execution, the D/A modules will ask for additional information to set up the D/A operation. It will also set the CPU at the highest hardware priority, i.e., it occupies the CPU. Thus, it will suspend other users' programs and stop the real-time clock. After the D/A operation, it will re-start other users' programs and the real-time clock.

The program listings are as follows:

```
C---    ************************************************************
C---    *                                                          *
C---    *                SPEECH RECORDING PROGRAM                  *
C---    *                MAIN     PROGRAM: DASP.FTN                *
C---    *                                                          *
C---    ************************************************************
C---
C---
C---    DATE: 4/#7/79
C---    MAWLIN YEH
C---
C---
C---    INDIRECT COMMAND FILE:  DA.CMD  --- EXECUTE D/A PROGRAM
C---
C---    MAIN PROGRAMS:  DASP.FTN        --- READ SPEECH DATA FILE
C---                    DARAMP.FTN      --- TEST D/A TERMINAL
C---    SUB PROGRAM:    DA64.MAC        --- 6.4KHZ D/A OUTPUT
C---                    DA8#.MAC        --- 8KHZ D/A OUTPUT
C---
C---    THIS PROGRAM WILL READ DATA FROM A DATA FILE AND THEN
C---    OUTPUT TO D/A CONVERTER.
C---
C---    THIS PROGRAM WILL SET CPU AT THE HIGHEST HARDWARE PRIORITY
C---    SO, IT WILL SUSPEND OTHER USERS' PROGRAMS AND STOP THE REAL-TIME CLOCK.
C---    AFTERWARE, THE REAL-TIME CLOCK WILL NOT BE ACCURATE.
C---
C---    THE PROCEDURE TO USE THIS FILE ARE AS FOLLOWS:
C---            1. LOGON PRIVILEGE UIC
C---            2. TYPE @DA
C---
C---    THE COMMAND FILE WILL DO THE FOLLOWING THINGS:
C---            1. IT WILL ASK WHICH FUNCTION ARE YOU GOING TO USE
C---               THERE ARE THREE FUNCTIOS IN THIS FILE:
C---                    A. RAMP FUNCTION -- TEST D/A & CHECK DELAY TIME.
C---                    B. 6.4KHZ D/A OUTPUT RATE
C---                    C. 8KHZ D/A OUTPUT RATE
C---            2. IT WILL PRODUCE SUITABLE TASK.
C---            3. THEN, IT WILL SET ONE DEVICE COMMON REGION CALLED
C---               COMMON.
C---            4. THEN, IT WILL INSTALL AND FIX THE PROPER TASK IN
C---               ORDER TO AVOID THE RELOCATION & THE USING OF CACHE
C---               MEMORY.
C---            5. EXECUTE THE TASK.
C---            6. AFTER EXECUTION, IT WILL REMOVE THE TASK AND
C---               ONE DEVICE COMMON REGION
C---
###1            INTEGER*2 FILNAM(16)
###2            COMMON /IBUF/IIS(1682#)
###3            COMMON /INF/NA,NB,NC
      C---      OPEN INPUT FILE
###4      11    TYPE *,' FILE NAME?'
###5            ACCEPT 2##,FILNAM
###6      2##   FORMAT(16A2)
###7            FILNAM(16)=#
###8            OPEN(UNIT=2,TYPE='OLD',READONLY,SHARED,NAME=FILNAM)
      C---      INPUT SPEECH DATA
```

```
FORTRAN IV-PLUS V02-51          11:22:30    31-MAR-80          PAGE 2
DASP.FTN        /TR:BLOCKS/WR

0009            READ(2,112)NSENT,IRATE,NSAMP,IUPPR,ILOWR,NTERMS,H
0010      112   FORMAT(6I5,1@X,4@A1)
0011            WRITE(5,312)NSENT,IRATE,NSAMP,IUPPR,ILOWR,NTERMS,H
0012      312   FORMAT(1X,6I5,1@X,4@A1)
0013            READ(2,111,END=88)IIS
0014      111   FORMAT(16I5)
        C---    ERROR CHECKING IN SIZE OF DATA
0015        88  IF(NSAMP .LE. 16820) GO TO 333
0016            TYPE *,' YOUR DATA FILE IS LARGER THAN DATA BUFFER !'
0017            TYPE *,' # OF SAMPLE =',NSAMP
0018            NSAMP=16820
0019            NF=1
        C---    ERROR CHECKING IN MAGNITUDE OF DATA
0020        333 DO 54 I=1,NSAMP
0021            IF(IIS(I) .GT. 2047) GO TO 334
0022            IF(IIS(I) .LT. -2048) GO TO 334
0023            GO TO 54
0024        334 NG=NG+1
0025            IF(NG .NE. 1) GO TO 335
0026            NF=1
0027            TYPE *,' DATA OUTSIDE RANGE OF D/A DATA REGISTER !'
0028        335 TYPE *,' IIS(',I,')=',IIS(I)
0029         54 CONTINUE
0030            IF(NF .EQ. 0) GO TO 22
0031            TYPE *,' DO YOU WANT CONTINUE [Y/N] ?'
0032            ACCEPT 336,Q
0033        336 FORMAT(A1)
0034            IF(Q .EQ. 'N') STOP
0035            NG=0
0036            NF=0
        C---    CONDITIONS
0037        22  TYPE *,' CHANNEL # ?'
0038            ACCEPT *,NCHNL
0039            NRT=0
        C---    OUTPUT DATA TO D/A
        C---    HAVE DELAY TIMING CONTROL
0040        12  CALL DA(NSAMP,NCHNL,NRT)
        C---    CONTINUE?
0041        13  TYPE *,'DO YOU WANT TO CONTINUE?'
0042            WRITE(5,*)'1=NEW FILE. 2=OUTPUT AGAIN. 3=STOP.'
0043            ACCEPT *,M
0044            IF(M .NE. 1) GO TO 33
0045            CLOSE(UNIT=2)
0046            GO TO 11
0047        33  IF(M .EQ. 2) GO TO 12
0048            IF(M .EQ. 3) STOP
0049            TYPE *,NA,NB,NC
0050            TYPE *,'?'
0051            GO TO 13
0052            END
```

```
C---    ************************************************************
C---    *                                                          *
C---    *              D/A CONVERTER TESTING PROGRAM               *
C---    *              MAIN PROGRAM: DARAMP.FTN                    *
C---    *                                                          *
C---    ************************************************************
C---
C---
C---    DATE: 3/25/79
C---    MAWLIN YEH
C---
C---
C---    MAIN PROGRAMS:  DASP.FTN        --- READ SPEECH DATA FILE
C---                    DARAMP.FTN      --- TEST D/A TERMINAL
C---    SUB PROGRAM:    DA64.MAC        --- 6.4KHZ D/A OUTPUT
C---                    DA8#.MAC        --- 8KHZ D/A OUTPUT
C---
C---    THIS PROGRAM WILL PRODUCE A SET OF DATA FROM A RAMP FUNCTION,
C---    AND THEN TRANSFER IT TO D/A CONVERTER.
C---
C---    THIS PROGRAM WILL SET CPU AT THE HIGHEST HARDWARE PRIORITY
C---    SO, IT WILL SUSPEND OTHER USERS' PROGRAM AND STOP THE REAL-TIME CLOCK.
C---    AFTERWARE, THE REAL-TIME CLOCK WILL NOT BE ACCURATE.
C---
C---    THE PROCEDURE TO USE THIS FILE ARE AS FOLLOWS:
C---            1. LOGON PRIVILEGE UIC
C---            2. TYPE @DA
C---
C---    THE COMMAND FILE WILL DO THE FOLLOWING THINGS:
C---            1. IT WILL ASK WHICH FUNCTION ARE YOU GOING TO USE
C---               THERE ARE THREE FUNCTIOS IN THIS FILE:
C---                    A. RAMP FUNCTION -- TEST D/A & CHECK DELAY TIME.
C---                    B. 6.4KHZ D/A OUTPUT RATE
C---                    C. 8KHZ D/A OUTPUT RATE
C---            2. IT WILL PRODUCE SUITABLE TASK.
C---            3. THEN, IT WILL SET ONE DEVICE COMMON REGION CALLED
C---               COMMON.
C---            4. THEN, IT WILL INSTALL AND FIX THE PROPER TASK IN
C---               ORDER TO AVOID THE RELOCATION & THE USING OF CACHE
C---               MEMORY.
C---            5. EXECUTE THE TASK.
C---            6. AFTER EXECUTION, IT WILL REMOVE THE TASK AND
C---               ONE DEVICE COMMON REGION
C---
0001            INTEGER*4 NHIGH
C---            COMMON BLOCKS
0002            COMMON /IBUF/IIS(1682#)
0003            COMMON /INF/NA,NB,NC
C---            CONDITIONS:
0004      33    TYPE *,' STARTING SAMPLE # ?'
0005            ACCEPT *,NSTART
0006            TYPE *,' END OF SAMPLE ?'
0007            ACCEPT *,NEND
0008            TYPE *,' SAMPLE INCREMENT ?'
0009            ACCEPT *,NSI
0010            TYPE *,' REPEAT # ?'
```

```
FORTRAN IV-PLUS V02-51          11:23:59    31-MAR-80           PAGE 2
DARAMP.FTN      /TR:BLOCKS/WR

0011            ACCEPT *,NRT
0012            TYPE *,' CHANNEL # ?'
0013            ACCEPT *,NCHNL
      C---      RAMP OR SQUARE FUNCTION
0014            TYPE *,' RAMP OR SQUARE ? ( 1=RAMP,2=SQUARE )'
0015            ACCEPT *,NN
0016            IF(NN .NE. 1) GO TO 30
      C---      PRODUCE RAMP DATA
0017            IK=0
0018            DO 40 I=NSTART,NEND,NSI
0019            IK=IK+1
0020            X=-1.+1./6400.*(I-1)
0021            X1=2047.*X
0022            X1=X1-0.5
0023            IF(X1 .LT. 0.) GO TO 40
0024            X1=X1+1.
0025      40    IIS(IK)=X1
0026            GO TO 55
      C---      PRODUCE SQUARE DATA
0027      30    TYPE *,' AMPLITUDE ?'
0028            ACCEPT *,NHIGH
0029            IK=0
0030            DO 44 I=NSTART,NEND,NSI
0031            IK=IK+1
0032      44    IIS(IK)=NHIGH
0033            DO 99 I=1,50
0034            IK=IK+1
0035      99    IIS(IK)=0
0036      55    IF(IK .GT. 16820) GO TO 400
      C---      OUTPUT DATA TO D/A
      C---      HAVE DELAY TIMING CONTROL
0037      12    CALL DA(IK,NCHNL,NRT)
      C---      CONTINUE?
0038      13    TYPE *,'DO YOU WANT TO CONTINUE?'
0039            WRITE(5,*)'1=NEW PARAMETERS. 2=OUTPUT AGAIN. 3=STOP.'
0040            ACCEPT *,M
0041            IF(M .EQ. 1) GO TO 33
0042            IF(M .EQ. 2) GO TO 12
0043            IF(M .EQ. 3) STOP
0044            TYPE *,NA,NB,NC
0045            TYPE *,'?'
0046            GO TO 13
0047      400   TYPE *,' SAMPLE # IS LARGER THAN BUFFER !'
0048            STOP
0049            END
```

```
 1                                      ;       ************************************************************
 2                                      ;       *                                                          *
 3                                      ;       *                  D/A OUTPUT PROGRAM                      *
 4                                      ;       *                  SUB PROGRAM: DA64.MAC                   *
 5                                      ;       *                                                          *
 6                                      ;       ************************************************************
 7                                      ;
 8                                      ;
 9                                      ;       DATE:3/25/79
10                                      ;       MAWLIN YEH
11                                      ;
12                                      ;       MAIN PROGRAM:   DASP.FTN
13                                      ;                       DARAMP.FTN
14                                      ;       SUB PROGRAM:    DA64.MAC
15                                      ;                       DA8#.MAC
16                                      ;
17                                      ;       THIS SUBROUTINE TRANSFERS DATA TO D/A DATA REGISTER
18                                      ;       IT USE ONE DEVICE COMMON BLOCK --- COMMON
19                                      ;
20                                      ;       THIS PROGRAM IS USED FOR THE ST-PDP PROGRAM CONTROLLED INTERFACE
21                                      ;       D/A DEVICE.
22                                      ;
23                                      ;
24                                              .TITLE SUB
25 ######                               ;       .PSECT
26                                              .GLOBL DA
27                                      ;       TRANSFER ARGUMENTS
28 ######  #125##                       DA:     MOV     (R5)+,R#        ; GET NUMBER OF ARGUMENTS
29 ######2 #127#1  ######'                      MOV     #SAMNO,R1       ; PSECT POINTER
30 ######6 #125#2                       1$:     MOV     (R5)+,R2        ; GET ADDR IN CALLING PROG.
31 ####1# #11221                                MOV     (R2),(R1)+      ; MOVE ARGS.
32 ####12 ##53##                                DEC     R#
33 ####14 1##374                                BPL     1$
34                                      ;       INITIALIZE PARAMETERS
35 ####16 #137#4  #####2'               START:  MOV     @#CHANEL,R4
36 ####22 #1#437  #####2'                       MOV     R4,@#LDAADR
37 ####26 #12737  ######  ####2#'              MOV     #0,@#LDADR      ; RESET DATA REGISTER
38 ####34 #137#1  ######'                       MOV     @#SAMNO,R1      ; # OF SAMPLES
39 ####4# #127#3  ######'                       MOV     #BUF1,R3        ; STARTING ADDRESS OF DATA
40 ####44 #137#2  ######4'                      MOV     @#REPEAT,R2     ; # OF REPEAT
41 ####5# 113737  #15776' ######'               MOVB    @#PSW,@#TEMP    ; STORE CURRENT PSW
42 ####56 112737  ###34#  #15776'               MOVB    #34#,@#PSW      ; SET THE HIGHEST HARDWARE PRIORITY
43                                      ;       OUTPUT LOOP
44 ####64 #1#437  #####2'               LOOP:   MOV     R4,@#LDAADR     ; CHANNEL ADDRESS
45 ####7# #12337  ####2#'                       MOV     (R3)+,@#LDADR   ; DATA OUTPUT
46 ####74 ###24#                                NOP
47 ####76 ###24#                                NOP
48 ###1## ###24#                                NOP
49 ###1#2 ###24#                                NOP
50 ###1#4 ###24#                                NOP
51 ###1#6 ###24#                                NOP
52 ###11# ###24#                                NOP
53 ###112 ###24#                                NOP
54 ###114 ###24#                                NOP
55 ###116 ###24#                                NOP
56 ###12# ###24#                                NOP
57 ###122 ###24#                                NOP
```

```
SUB     MACRO M111#  31-MAR-8# 11:26  PAGE 1-1


      5# ###124  ###24#                    NOP
      59 ###126  ###24#                    NOP
      6# ###13#  ###24#                    NOP
      61 ###132  ###24#                    NOP
      62 ###134  ###24#                    NOP
      63 ###136  ###24#                    NOP
      64 ###14#  ###24#                    NOP
      65 ###142  ###24#                    NOP
      66 ###144  ###24#                    NOP
      67 ###146  ###24#                    NOP
      68 ###15#  ###24#                    NOP
      69 ###152  ###24#                    NOP
      7# ###154  ###24#                    NOP
      71 ###156  ###24#                    NOP
      72 ###16#  ###24#                    NOP
      73 ###162  ###24#                    NOP
      74 ###164  ###24#                    NOP
      75 ###166  ###24#                    NOP
      76 ###17#  ###24#                    NOP
      77 ###172  ###24#                    NOP
      78 ###174  ###24#                    NOP
      79 ###176  ###24#                    NOP
      8# ###2##  ###24#                    NOP
      81 ###2#2  ###24#                    NOP
      82 ###2#4  ###24#                    NOP
      83 ###2#6  ###24#                    NOP
      84 ###21#  ###24#                    NOP
      85 ###212  ###24#                    NOP
      86 ###214  ###24#                    NOP
      87 ###216  ###24#                    NOP
      88 ###22#  ###24#                    NOP
      89 ###222  ###24#                    NOP
      9# ###224  ###24#                    NOP
      91 ###226  ###24#                    NOP
      92 ###23#  ###24#                    NOP
      93 ###232  ###24#                    NOP
      94 ###234  ###24#                    NOP
      95 ###236  ###24#                    NOP
      96 ###24#  ###24#                    NOP
      97 ###242  ###24#                    NOP
      98 ###244  ###24#                    NOP
      99 ###246  ###24#                    NOP
     1## ###25#  ###24#                    NOP
     1#1 ###252  ###24#                    NOP
     1#2 ###254  ###24#                    NOP
     1#3 ###256  ###24#                    NOP
     1#4 ###26#  ###24#                    NOP
     1#5 ###262  ###24#                    NOP
     1#6 ###264  ###24#                    NOP
     1#7 ###266  ###24#                    NOP
     1#8 ###27#  ###24#                    NOP
     1#9 ###272  ###24#                    NOP
     11# ###274  ###24#                    NOP
     111 ###276  ###24#                    NOP
     112 ###3##  ###24#                    NOP
     113 ###3#2  ###24#                    NOP
     1[redacted]                           NOP
```

```
 115 ###3#6  ###24#                          NOP
 116 ###31#  ###24#                          NOP
 117 ###312  ###24#                          NOP
 118 ###314  ###24#                          NOP
 119 ###316  ###24#                          NOP
 12# ###32#  ###24#                          NOP
 121 ###322  ###24#                          NOP
 122 ###324  ###24#                          NOP
 123 ###326  ###24#                          NOP
 124 ###33#  ###24#                          NOP
 125 ###332  ###24#                          NOP
 126 ###334  ###24#                          NOP
 127 ###336  ###24#                          NOP
 128 ###34#  ###24#                          NOP
 129 ###342  ##53#1                          DEC     R1              ; END OF DATA?
 13# ###344  1##247                          BPL     LOOP
 131 ###346  #137#1  ######'                 MOV     @#SAMNO,R1
 132 ###352  #1#437  #####2'                 MOV     R4,@#LDAADR     ; SET CHANNEL #
 133 ###356  #12737  ######  ####2#'        MOV     #0,@#LDADR      ; RESET D/A
 134 ###364  #127#3  ######'                 MOV     #BUF1,R3        ; STARTING ADDRESS OF DATA
 135 ###37#  ##53#2                          DEC     R2
 136 ###372  1##234                          BPL     LOOP            ; REPEAT
 137 ###374  113737  ######' #15776'         MOVB    @#TEMP,@#PSW    ; RESTORE PSW
 138 ###4#2  ###2#7                          RTS     PC              ; TRANSFER FINISHED
 139                                ; DATA BUFFER
 14# ######                                  .PSECT IBUF,D,OVR,GBL
 141                                         .GLOBL BUF1
 142 ######                         BUF1:    .BLKW   1682#.
 143                                ;        TEMPERARY STORAGE SPACE
 144 ######                                  .PSECT DATA,D,OVR,GBL
 145                                         .GLOBL TEMP
 146 ######                         TEMP:    .BLKW 1         ; TEMP. STORAGE FOR PSW
 147                                ; INFORMATION TRANSFER
 148 ######                                  .PSECT INF,D,OVR,GBL
 149                                         .GLOBL SAMNO,CHANEL,REPEAT
 15# ######                         SAMNO:   .BLKW 1
 151 #####2                         CHANEL:  .BLKW 1
 152 #####4  ######                 REPEAT:  .WORD #
 153                                ; DEVICE COMMON BLOCK
 154 ######                                  .PSECT COMMON,D,OVR,GBL
 155 ######                         LSR:     .BLKW 1
 156 #####2                         LDAADR:  .BLKW 1
 157 #####4                                  .BLKW 6
 158 ####2#                         LDADR:   .BLKW 1
 159 ###22                                   .BLKW 27
 16# ##1##                                   .BLKW 352#.
 161 #157##                                  .BLKW 31.
 162 #15776                         PSW:     .BLKW 1
 163         #####1                          .END
```

```
                                ;       ************************************************************
                                ;       *                                                          *
                                ;       *                D/A OUTPUT PROGRAM                        *
                                ;       *                SUB PROGRAM: DA8#.MAC                     *
                                ;       *                                                          *
                                ;       ************************************************************
                                ;
                                ;
                                ;       DATE:3/26/79
                                ;       MAWLIN YEH
                                ;
                                ;       MAIN PROGRAM:   DASP.FTN
                                ;                       DARAMP.FTN
                                ;       SUB PROGRAM:    DA64.MAC
                                ;                       DA8#.MAC
                                ;
                                ;       THIS SUBROUTINE TRANSFERS DATA TO D/A DATA REGISTER
                                ;       IT USE ONE DEVICE COMMON BLOCK --- COMMON
                                ;
                                ;       THIS PROGRAM IS USED FOR THE ST-PDP PROGRAM CONTROLLED INTERFACE
                                ;       D/A DEVICE.
                                ;
                                ;
                                        .TITLE SUB
000000                          .PSECT
                                .GLOBL DA
                                ;       TRANSFER ARGUMENTS
000000  012500          DA:     MOV     (R5)+,R0        ; GET NUMBER OF ARGUMENTS
000002  012701  000000'         MOV     @SAMNO,R1       ; PSECT POINTER
000006  012502          10:     MOV     (R5)+,R2        ; GET ADDR IN CALLING PROG.
000010  011221                  MOV     (R2),(R1)+      ; MOVE ARGS.
000012  005300                  DEC     R0
000014  100374                  BPL     10
                                ;       INITIALIZE PARAMETERS
000016  013704  000002' START:  MOV     @@CHANEL,R4
000022  010437  000002'         MOV     R4,@@LDAADR
000026  012737  000000  000020' MOV     #0,@@LDADR      ; RESET DATA REGISTER
000034  013701  000000'         MOV     @@SAMNO,R1      ; # OF SAMPLES
000040  012703  000000'         MOV     #BUF1,R3        ; STARTING ADDRESS OF DATA
000044  013702  000004'         MOV     @@REPEAT,R2     ; # OF REPEAT
000050  113737  015776  000000' MOVB    @@PSW,@@TEMP    ; STORE CURRENT PSW
000056  112737  000340  015776' MOVB    #340,@@PSW      ; SET THE HIGHEST HARDWARE PRIORITY
                                ; OUTPUT LOOP
000064  010437  000002' LOOP:   MOV     R4,@@LDAADR     ; CHANNEL ADDRESS
000070  012337  000020'         MOV     (R3)+,@@LDADR   ; DATA OUTPUT
000074  000240                  NOP
000076  000240                  NOP
000100  000240                  NOP
000102  000240                  NOP
000104  000240                  NOP
000106  000240                  NOP
000110  000240                  NOP
000112  000240                  NOP
000114  000240                  NOP
000116  000240                  NOP
000120  000240                  NOP
000122  000240                  NOP
```

```
SUB     MACRO M1111  31-MAR-88 11:28  PAGE 1-1


        58 000124  000240                       NOP
        59 000126  000240                       NOP
        60 000130  000240                       NOP
        61 000132  000240                       NOP
        62 000134  000240                       NOP
        63 000136  000240                       NOP
        64 000140  000240                       NOP
        65 000142  000240                       NOP
        66 000144  000240                       NOP
        67 000146  000240                       NOP
        68 000150  000240                       NOP
        69 000152  000240                       NOP
        70 000154  000240                       NOP
        71 000156  000240                       NOP
        72 000160  000240                       NOP
        73 000162  000240                       NOP
        74 000164  000240                       NOP
        75 000166  000240                       NOP
        76 000170  000240                       NOP
        77 000172  000240                       NOP
        78 000174  000240                       NOP
        79 000176  000240                       NOP
        80 000200  000240                       NOP
        81 000202  000240                       NOP
        82 000204  000240                       NOP
        83 000206  000240                       NOP
        84 000210  000240                       NOP
        85 000212  000240                       NOP
        86 000214  000240                       NOP
        87 000216  000240                       NOP
        88 000220  000240                       NOP
        89 000222  000240                       NOP
        90 000224  000240                       NOP
        91 000226  000240                       NOP
        92 000230  000240                       NOP
        93 000232  000240                       NOP
        94 000234  000240                       NOP
        95 000236  000240                       NOP
        96 000240  000240                       NOP
        97 000242  000240                       NOP
        98 000244  000240                       NOP
        99 000246  000240                       NOP
       100 000250  000240                       NOP
       101 000252  000240                       NOP
       102 000254  000240                       NOP
       103 000256  000240                       NOP
       104 000260  000240                       NOP
       105 000262  000240                       NOP
       106 000264  000240                       NOP
       107 000266  000240                       NOP
       108 000270  000240                       NOP
       109 000272  000240                       NOP
       110 000274  000240                       NOP
       111 000276  000240                       NOP
       112 000300  005301   DEC     R1          ; END OF DATA?
       113 000302  100270   BPL     LOOP
       114 000304  013701   000000' MOV     @#SAMNO,R1
```

```
SUB     MACRO M111#  31-MAR-8# 11:28  PAGE 1-2

    115 ###31#  #1#437  ######2'                MOV     R4,@#LDAADR     ; SET CHANNEL #
    116 ###314  #12737  ######  #####2#'        MOV     #0#,@#LDADR     ; RESET D/A
    117 ###322  #127#3  ######'                 MOV     #BUF1,R3        ; STARTING ADDRESS OF DATA
    118 ###326  ##53#2                          DEC     R2
    119 ###33#  1##256                          BPL     LOOP            ; REPEAT
    12# ###332  113737  ######' #15776'         MOVB    @#TEMP,@#PSW    ; RESTORE PSW
    121 ###34#  ###2#7                          RTS     PC              ; TRANSFER FINISHED
    122                             ; DATA BUFFER
    123 ######                                  .PSECT IBUF,D,OVR,GBL
    124                                         .GLOBL BUF1
    125 ######                      BUF1:       .BLKW   1682#.
    126                             ;           TEMPERARY STORAGE SPACE
    127 ######                                  .PSECT DATA,D,OVR,GBL
    128                                         .GLOBL TEMP
    129 ######                      TEMP:       .BLKW 1         ; TEMP. STORAGE FOR PSW
    13#                             ; INFORMATION TRANSFER
    131 ######                                  .PSECT INF,D,OVR,GBL
    132                                         .GLOBL SAMNO,CHANEL,REPEAT
    133 ######                      SAMNO:      .BLKW 1
    134 #####2                      CHANEL:     .BLKW 1
    135 #####4  ######              REPEAT:     .WORD #
    136                             ; DEVICE COMMON BLOCK
    137 ######                                  .PSECT COMMON,D,OVR,GBL
    138 ######                      LSR:        .BLKW 1
    139 #####2                      LDAADR:     .BLKW 1
    14# #####4                                  .BLKW 6
    141 ####2#                      LDADR:      .BLKW 1
    142 ####22                                  .BLKW 27
    143 ###1#8                                  .BLKW 352#.
    144 #157##                                  .BLKW 31.
    145 #15776                      PSW:        .BLKW 1
    146         #####1                          .END
```

```
.MAIN.  MACRO M111#  31-MAR-8# 11:31  PAGE 1

                                ;       ********************************************************
                                ;       *                                                      *
                                ;       *                DEVICE COMMON BLOCK                   *
                                ;       *                                                      *
                                ;       ********************************************************
                                ;
                                ;       THIS PROGRAM IS USED TO SET A DEVICE COMMON BLOCK.
                                ;
                                        .PSECT  COMMON,RW,D,GBL,REL,OVR
                                LSR:    .BLKW   1
                                LDAADR: .BLKW   1
                                RADDR:  .BLKW   1
                                LADADR: .BLKW   1
                                        .BLKW   4
                                LDADR:  .BLKW   1
                                LMADR:  .BLKW   1
                                LWDCT:  .BLKW   1
                                        .BLKW   25
                                        .BLKW   352#.
                                        .BLKW   31.
                                PSW:    .BLKW   1
                                RSR=LSR
                                RADADR=LADADR
                                RMADR=LMADR
                                        .END
```

```
!       ********************************************************
!       *                                                      *
!       *       INDIRECT COMMAND FILE FOR D/A CONVERTING       *
!       *                                                      *
!       ********************************************************
!
!       THIS COMMAND IS USED TO RUN THE D/A PROGRAM.
!       YOU HAVE TO USE THE PRIVILEGE UIC.
!       SO,BECAREFUL!
!       WHEN YOU EXECUTE THIS FILE, DON'T DO OTHER ACTIONS
!       UNTIL IT FINISHS.
SET /MAIN=COMMON:762#:16#:DEV
INS DK:[1,1]COMMON
!       COMMON DEVICE REGION HAS BEEN SET UP.
.ENABLE SUBSTITUTION
.SETF A1
.SETF A2
.SETF A3
.SETF A4
.SETF A5
.SETF A6
.SETF A7
.SETF A8
!
!       THERE ARE FOUR CHOICES:
!       1. 6.4KHZ SPEECH
!       2. 8.#KHZ SPEECH
!       3. 6.4KHZ RAMP
!       4. 8.#KHZ RAMP
.2:     .ASK A 6.4KHZ SPEECH
        .IFF A .GOTO 1#
        .IFT A1 .GOTO 2#
        .SETT A1
.WAIT FOR
FOR DASP=DASP
.2#:    .IFT A2 .GOTO 3#
        .SETT A2
.WAIT MAC
MAC DA64=DA64
.3#:    .SETS B1 "DASP64"
        .IFT A3 .GOTO 2##
        .SETT A3
        .SETS B2 "DA64"
        .SETS B3 "DASP"
        .GOTO 1##
.1#:    .ASK C 8.#KHZ SPEECH
        .IFF C .GOTO 5#
        .IFT A1 .GOTO 6#
        .SETT A1
.WAIT FOR
FOR DASP=DASP
.6#:    .IFT A4 .GOTO 7#
        .SETT A4
.WAIT MAC
MAC DA8#=DA8#
.7#:    .SETS B1 "DASP8#"
        .IFT A5 .GOTO 2##
        .SETT A5
        .SETS B2 "DA8#"
        .SETS B3 "DASP"
        .GOTO 1##
.5#:    .ASK D 6.4KHZ RAMP
        .IFF .GOTO
```

```
        .IFT A6 .GOTO 9#
        .SETT A6
.WAIT FOR
FOR DARAMP=DARAMP
.9#:    .IFT A2 .GOTO 91
        .SETT A2
.WAIT MAC
MAC DA64=DA64
.91:    .SETS B1 "DARM64"
        .IFT A7 .GOTO 2##
        .SETT A7
        .SETS B2 "DA64"
        .SETS B3 "DARAMP"
        .GOTO 1##
.8#:    .ASK E 8.#KHZ RAMP
        .IFF E .GOTO 3##
        .IFT A6 .GOTO 81
        .SETT A6
.WAIT FOR
FOR DARAMP=DARAMP
.81:    .IFT A4 .GOTO 82
        .SETT A4
.WAIT MAC
MAC DA8#=DA8#
.82:    .SETS B1 "DARM8#"
        .IFT A8 .GOTO 2##
        .SETT A8
        .SETS B2 "DA8#"
        .SETS B3 "DARAMP"
.1##:   .OPEN DA##1.CMD
.ENABLE DATA
'B1'/PR:#='B2','B3'
/
COMMON=COMMON:RW
//
.DISABLE DATA
.CLOSE
.WAIT TKB
TKB @DA##1
.2##:   ;       THE FOLLOWING STEPS HAVE TO BE DONE CAREFULLY.
;       IF YOU ARE READY, PLEASE TYPE 'RES ...AT.'
;       OR YOU CAN TYPE 'ABO ...AT.' TO ABORT THIS COMMAND FILE.
.PAUSE DA
INS 'B1'
FIX 'B1'
;       COMMAND WILL BE PAUSED RIGHT HERE.
;       AFTER EXECUTION OF D/A FILE, PLEASE TYPE 'RES ...AT.'
;       TO REMOVE THE TASK.
RUN 'B1'
.PAUSE DA
REM 'B1'
.3##:   .ASK F DO YOU WANT TO RUN IT AGAIN
        .IFT F .GOTO 2
SET /NOMAIN=COMMON
PIP DA##1.CMD;*/DE
.IFF A1 .GOTO 3#1
.WAIT PIP
PIP DASP.OBJ;*/DE
.3#1:   .IFF A2 .GOTO 3#2
.WAIT PIP
PIP DA64.OBJ;*/DE
.3#2:   .IFF A3 .GOTO 3#3
.WAIT PIP
PIP DASP64.TSK;*/DE
```

```
.3#3:    .IFF A4 .GOTO 3#4
.WAIT PIP
PIP DA##.OBJ;*/DE
.3#4:    .IFF A5 .GOTO 3#5
.WAIT PIP
PIP DASP##.TSK;*/DE
.3#5:    .IFF A6 .GOTO 3#6
.WAIT PIP
PIP DARAMP.OBJ;*/DE
.3#6:    .IFF A7 .GOTO 3#7
.WAIT PIP
PIP DARM64.TSK;*/DE
.3#7:    .IFF A8 .GOTO 3#8
.WAIT PIP
PIP DARM8#.TSK;*/DE
.3#8:    ;       END.
```

END

DATE
FILMED

8-80

DTIC